

AD/A-003 486

STRUCTURED DESCRIPTIONS OF COMPLEX
CURVED OBJECTS FOR RECOGNITION AND
VISUAL MEMORY

Ramakant Nevatia

Stanford University

Prepared for:

Advanced Research Projects Agency

October 1974

DISTRIBUTED BY:

NTIS

National Technical Information Service
U. S. DEPARTMENT OF COMMERCE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER STAN-CS-74-464	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER AD/A-003486
4. TITLE (and Subtitle) STRUCTURED DESCRIPTIONS OF COMPLEX CURVED OBJECTS FOR RECOGNITION AND VISUAL MEMORY.		5. TYPE OF REPORT & PERIOD COVERED technical, Oct. 1974
7. AUTHOR(s) Ramakant Nevatia		6. PERFORMING ORG. REPORT NUMBER STAN-CS-74-464
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Science Dept. Stanford University Stanford, California 94305		8. CONTRACT OR GRANT NUMBER(s) DAHC 15-73-c-0435
11. CONTROLLING OFFICE NAME AND ADDRESS ARPA/IPT Attn: S. D. Crocker 1400 Wilson Blvd.; Arlington, Va. 22209		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) ONR Representative: Philip Surra Durand Aeronautics Bldg., Rm. 165 Stanford University Stanford, California 94305		12. REPORT DATE October, 1974
		13. NUMBER OF PAGES 128
		15. SECURITY CLASS. (of this report) Unclassified
16. DISTRIBUTION STATEMENT (of this Report) Releasable without limitations on dissemination.		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Reproduced by NATIONAL TECHNICAL INFORMATION SERVICE US Department of Commerce Springfield, VA. 22151		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) Description and recognition of three-dimensional objects from range data obtained by a laser triangulation technique are described. A complex object is described by decomposition into subparts and relations of these sub-parts. The individual parts are described by <u>generalized cones</u> , which are defined by a space curve known as the axis, and arbitrary shaped normal cross-sections along this axis. Techniques for segmenting an object into sub-parts and generating		

(continued)

FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

structured, symbolic, graph like descriptions are described. These symbolic descriptions are matched with stored descriptions and the best match is picked for recognition. A limited amount of indexing capability exists to efficiently retrieve a sub-class of similar objects from the models stored in the memory. Indexing is a necessity if a large number of visual models is to be used.

Results of working programs for the stated tasks on many actual scenes are presented. The scenes consist of single as well as multiple objects.

This work was submitted to the Department of Electrical Engineering and the Committee on Graduate Studies of Stanford University in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

il
OCTOBER 1974

STAN-CS-74-464

STRUCTURED DESCRIPTIONS OF COMPLEX CURVED OBJECTS FOR RECOGNITION AND VISUAL MEMORY

by

Ramakant Nevatia

Description and recognition of three-dimensional objects from range data obtained by a laser triangulation technique are described. A complex object is described by decomposition into sub-parts and relations of these sub-parts. The individual parts are described by *generalized cones*, which are defined by a space curve known as the axis, and arbitrary shaped normal cross-sections along this axis.

Techniques for segmenting an object into sub-parts and generating structured, symbolic, graph like descriptions are described. These symbolic descriptions are matched with stored descriptions and the best match is picked for recognition. A limited amount of *indexing* capability exists to efficiently retrieve a sub-class of similar objects from the models stored in the memory. Indexing is a necessity if a large number of visual models is to be used.

Results of working programs for the stated tasks on many actual scenes are presented. The scenes consist of single as well as multiple objects.

This work was submitted to the Department of Electrical Engineering and the Committee on Graduate Studies of Stanford University in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

The research reported here was supported in part by the Advanced Research Projects Agency of the Department of Defense under Contract DAHC-15-73-C-0435.

The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or of the U. S. Government.

Reproduced in the USA. Available from the National Technical Information Service, Springfield, Virginia 22151.

ACKNOWLEDGEMENTS

I am grateful to my advisor, Dr. Thomas O. Binford, for his constant guidance and many helpful insights. He provided numerous suggestions and many ideas used in this work developed in discussions with him.

I would like to thank Prof. Jerome Feldman for his reading of this thesis and taking interest in its progress; and Prof. Bernard Widrow for his attempts at making it more readable. I would like to thank Prof. John McCarthy for providing the opportunity to work at the Stanford Artificial Intelligence Laboratory and make this thesis possible across departmental lines.

An experimental thesis of this nature would not have been possible without drawing on many other peoples's support. In particular, I wish to acknowledge the following: Dr. Gerald J. Agin, for his data acquisition programs and some picture files; Alan Borning for his preliminary processing programs; Wing Hui for help with the programming of the "Projection" routines; and Joe Zingheim and Karl Pingle for their support of the "Hand/Eye" hardware and software.

TABLE OF CONTENTS

SECTION	PAGE
1.0 INTRODUCTION	I
1.1 AN OVERVIEW	3
2.0 REPRESENTATION	8
2.1 GENERALIZED CONE PRIMITIVES	8
2.2 OTHER PRIMITIVES	11
3.0 DATA ACQUISITION AND BOUNDARY ORGANIZATION	12
3.1 LASER TRIANGULATION RANGING	12
3.2 BOUNDARY ORGANIZATION	16
3.3 BODY SEPARATION	18
4.0 BODY SEGMENTATION AND PRIMITIVE DESCRIPTIONS	22
4.1 CONE DEFINITIONS	23
4.2 LOCAL CONE SAMPLES: METHOD OF PROJECTIONS	24
4.3 EXTENSION OF LOCAL CONES	25
4.4 OTHER CONE DESCRIPTION METHODS	34
4.5 CHOICE OF SEGMENTATIONS	37
4.6 OTHER DESCRIPTIONS OF PRIMITIVES	43
5.0 SYMBOLIC DESCRIPTIONS	44
5.1 PIECE DESCRIPTIONS	46
5.2 LINKING OF PIECES AND JOINT DESCRIPTIONS	47
5.3 OBJECT DESCRIPTION	51
5.4 LINKING OF SHADOWED PARTS	52
5.5 JOINT/PART SEMANTICS	53
6.0 RECOGNITION	54
6.1 AN EXAMPLE	58
6.2 MATCHING WITH A MODEL	61
6.3 MATCHING OF OBJECT PIECES	64
6.4 OCCLUSION AND SHADOWS	67
6.5 SELECTION OF A MATCH	69
6.6 VERIFICATION	70
6.7 INDEXING INTO VISUAL MEMORY	71
6.8 MODEL ACQUISITION AND LEARNING	74
7.0 RESULTS AND CONCLUSIONS	76
7.1 RESULTS	76
7.2 EXECUTION TIMES AND MEMORY REQUIREMENTS	90
7.3 FURTHER RESEARCH	92
7.4 CONCLUSIONS	93
APPENDIX 1 BOUNDARY ORGANIZATION	96
APPENDIX 2 DESCRIPTION OF ALGORITHMS	101
APPENDIX 3 DATA STRUCTURES	116
REFERENCES	118

LIST OF ILLUSTRATIONS

FIGURE	PAGE
1.1 Block Diagram	4
2.1 Segmentation of a Human Form into Sub-parts	9
3.1 Schematic View of Triangulation Ranging	13
3.2 Schematic of the Ranging Apparatus	14
3.3 Laser Scans for a Doll	15
3.4 Boundary Constructed from the Scans in Fig. 3.3	17
3.5 A TV Picture of a Doll and a Snake	19
3.6 Laser Scans for the Scene of Fig. 3.5	20
3.7 Boundary Derived from the Scans of Fig. 3.6	21
4.1 Two choices for Axis Points	24
4.2 Cross-sections from Projection in a Chosen Direction	26
4.3 Local Cones Generated from Fig. 4.2	27
4.4 Local Cones Generated by Projection in Eight Directions	28
4.5 Extension of a Cone	30
4.6 Axes of Extended Cones for the Doll of Fig. 4.2	31
4.7 Cones for a Hammer	32
4.8 Laser Scans for a Hammer	32
4.9 Cone Descriptions for a Glove	35
4.10 Cone Descriptions for a Horse	36
4.11 Different Cones for a Rectangle	38
4.12 Selected Cones from fig. 4.6 for a Doll	39
4.13 All Extended Cones for Scene of Fig. 3.5	41
4.14 Selected Cones from Fig. 4.13	42
5.1 Connection Graph of a Doll	45
5.2 Joints between Pieces of the Doll of fig. 4.12	48
5.3 Different Joint Types	50
6.1 Another View of the Glove of Fig. 4.9	55
6.2 Stick Figures of Two Objects	57
6.3 A View of a Doll	59
6.4 Connection Graph of the Doll in Fig. 6.3	60
6.5 Matching Results for the Doll of fig. 6.3	62
7.1 A TV Picture of a Horse and a Ring	77
7.2 A TV Picture of a Horse and a Doll	77
7.3 Segmentation of Scene Corresponding to Fig. 7.1	78
7.4 Recognition Results for Horse in Fig. 7.3	79
7.5 Segmentation of Scene Corresponding to Fig. 7.2	80
7.6 A View of a Horse	81
7.7 Recognition Results for Horse of Fig. 7.6	82
7.8 A Snake	83
7.9 Another View of a Doll	84
7.10 Recognition Results for the Doll in fig. 7.9	84
7.11 A Glove	85
7.12 Recognition Results for the Glove in Fig. 7.11	85

LIST OF ILLUSTRATIONS

7.13 Another View of a Doll	86
7.14 Another Hammer	86
7.15 Another View of a Horse	87
A1.1 Boundary Linking for Different Cases	97
A1.2 Examples of Segment Overlaps and Non-overlaps	99
A2.1 Cross-sections for a Selected Projection Direction.	102
A2.2 Axis Refinement	104

CHAPTER I

INTRODUCTION

The research described in this thesis is part of a continuing effort, at the Stanford Hand-eye project, to develop the capabilities for a machine to analyze scenes of complex objects and manipulate these objects for tasks such as part assembly. Much of the past work in three-dimensional scene analysis has concentrated on scenes containing polyhedral objects only. This thesis is concerned with machine generation of symbolic descriptions for three-dimensional complex, curved objects and their recognition based on these descriptions. The complexity of the objects viewed is typified by toy animals such as a horse and a doll, and by hand tools such as a hammer. (The reader may wish to glance through the figures in chapter 7 for a sampling of the scenes these programs work with.) Our concern here will be with the *shape* properties of an object only. Other cues such as color and surface texture have not been used.

Previous Work:

The problem of object recognition has received extensive attention in the literature on Pattern Recognition ([Duda]), though the emphasis has been on the recognition of two-dimensional patterns. Analysis of three-dimensional scenes from their two-dimensional camera images presents the following difficulties: the two-dimensional image of the object changes with the viewing angle; when multiple objects in a scene occlude each other, only parts of some objects will be seen in the camera image, and also the occluding objects need to be separated from each other. A non-convex object can partially occlude itself. Additionally, in our system we have allowed parts of an object to be articulated (*i.e.* move with respect to the other parts). The classical pattern recognition methods have not been concerned with such variations and have only considered statistical variations of a fixed pattern.

A popular paradigm in pattern recognition has been that of *Template Matching*. Template matching consists of matching an input pattern with a model pattern, known as a template, on a point to point basis. The matching is usually performed at the level of input measurements, *e.g.* the intensity levels in the image or the values in a range matrix. A simple metric, such as the root mean square of the differences, or the correlation of the image and the template establishes the quality of the match. Such template matching is directly applicable only if the image of the entire scene is invariant, *e.g.* for two-dimensional patterns. Some flexible template matching schemes have been suggested ([Widrow],[Fischler]). Parts of such a template are allowed to be moved with respect to the others. Comparison of the observed scene with such a template finds the best "distortion" of the template required to match with the scene. These techniques, utilizing point to point matching of the model pattern and the scene are difficult to extend for the expected variations of three-dimensional scenes. Further, template matching does not provide useful similarity and difference descriptions, such as two objects are similar but for a missing limb in one.

The early work on three-dimensional scene analysis simplified the problem by restricting to homogeneous polyhedral objects. In a now classical work, Roberts ([Roberts 63]) extracted edge information from simple polyhedral scenes and compared the resulting descriptions with possible projections of stored models for object recognition. With multiple objects in the scene, many combinations of known models were tried. It is clear that for an increasing number of models, these techniques soon become computationally infeasible.

The attention then turned to the problem of "Body separation", i.e. separation of occluding bodies in a scene (See [Guzman], [Falk], and [Waltz]). Grape ([Grape]) combined the separation of bodies with recognition, by removing parts of the scene recognized as belonging to a known body. All of these techniques were designed to work with polyhedral objects only, and extensively use the properties of edges and vertices. Though some impressive results have been reported ([Waltz], [Grape]), and perhaps some useful abstractions can be made, the specific techniques used fail to generalize to a wider class of objects.

Among previous work on curved objects, B.K.P. Horn ([Horn]) presented techniques for extracting three dimensional depth data from a TV image, using reflection characteristics of the surface. Krakauer ([Krakauer]) represented objects by connections of brightness contours. Ambler *et al* ([Ambler]) describe experiments with simple shapes, including curved objects, using relations within a two-dimensional image. However, none of these efforts really addresses the problem of "shape" representation and description. Work on outdoor scene analysis is also concerned with non-polyhedral objects ([Bajcsy], [Yakimovsky]), but again no attention has been paid to shape analysis.

Our work is based on initial work of G.J. Agin and T.O. Binford ([Agin 72, 73], [Binford]). Binford proposed a new representation for complex objects by segmentation into primitive parts described as Generalized Cylinders (and cones), which are defined by a space curve, known as the axis, and a set of cross-sections along this curve. The shape and the size of the cross-sections may change continuously along the axis. Agin built a laser ranging system to measure the three-dimensional positions of the points on an object surface. The 3-d position information helps resolve ambiguities caused by occlusion. (This system only measures the 3-d positions of points on the surface visible to the camera.)

Agin described preliminary efforts at generating descriptions from the three-dimensional range data. However, these description techniques were unstructured; only isolated part descriptions were generated and not related to each other to make up a complete body. Further, the description of individual parts had some major deficiencies. In particular, some descriptions merged nearby but distinct parts. In this thesis, we present new description techniques that are different conceptually and in implementation. They generate adequate segmentation and part descriptions for an object and are a major advance over the previous work. The segmentation techniques are general and work without a priori knowledge of the the object being viewed. Structured, symbolic descriptions are generated based on these segmentations.

Approach:

The techniques described here use the same representation and laser ranging system. These are briefly described in chapter 2 and section 3.1, to allow an independent reading of this thesis. The remainder of this thesis represents the author's own contributions. (Note that this thesis consistently uses the first person plural.)

The chosen representation is designed to cope with the problems of 3-d scene analysis mentioned earlier. The major component of the chosen representation is the *Structure* of the object, defined by the connectivity pattern of its sub-parts. This structure is invariant with the viewing angles, except for the absence of some parts in a particular view due to occlusion (computation of this structure from certain viewing angles may be difficult). However, some objects are reasonably described as having alternate structures (details in chapter 6). In such cases,

we just store the alternative descriptions of the same model; each description is examined independently for recognition. The expected number of such alternates is small. Articulation of a limb is easily described by its relation to other limbs. Our recognition procedures use descriptions generated from the observed data in terms of this representation.

Two descriptions are matched in their structure as well as the details of the sub-parts. Note that since we have segmented descriptions of the scene, the matching proceeds directly and does not have to try various "distortions" of the model description. Recognition is by picking the model which matches best with the observed description. Our system has a limited amount of *indexing* capability, i.e. a list of similar objects can be retrieved from the memory using the descriptions of the current objects, and comparison with each known model is not necessary. Models for recognition are obtained by storing machine generated descriptions of the objects. Such a structure of visual models is known as a *Visual Memory*.

Among the contributions of this thesis are: the techniques for segmenting the object into sub-parts from the observed data; the structure of the symbolic descriptions and techniques for generating such descriptions; and methods for efficient recognition from these descriptions including indexing. Working programs for the presented techniques have been written. (All of the described programs run without human intervention.)

In the next section, we present an overview of our methods and discuss the adequacy of our techniques.

1.1 AN OVERVIEW

The conventional input for computer vision programs has been the output of a TV camera or a digitized photograph. A camera image is two dimensional, whereas the space viewed is three-dimensional. The picture information is incomplete in the sense that the depth of the points in the image cannot be directly inferred. We use a laser triangulation ranging method that gives us direct three-dimensional information about the points in the image; this method is briefly described in chapter 3.

Representation of an object by segmenting it into simpler sub-parts represented as generalized cones is discussed in chapter 2. Primitives other than generalized cones are also suggested but have not been used in our system. Each sub-part will also be referred to as a *Piece*; various pieces connect at a *joint*. The connectivity of the sub-parts of an object defines the *structure* of the object.

Techniques:

The block diagram of Fig. 1.1, describes schematically the processing of the range data. Following is an overview of these processes.

Construction of the boundaries of the objects in the scene has been found to be useful in structuring the processing of the surface range data in our system. Depth discontinuities are used to determine object boundaries, and correspond to the normal notion of object boundaries. The ranging method provides us with an outer boundary that is not sensitive to gray level

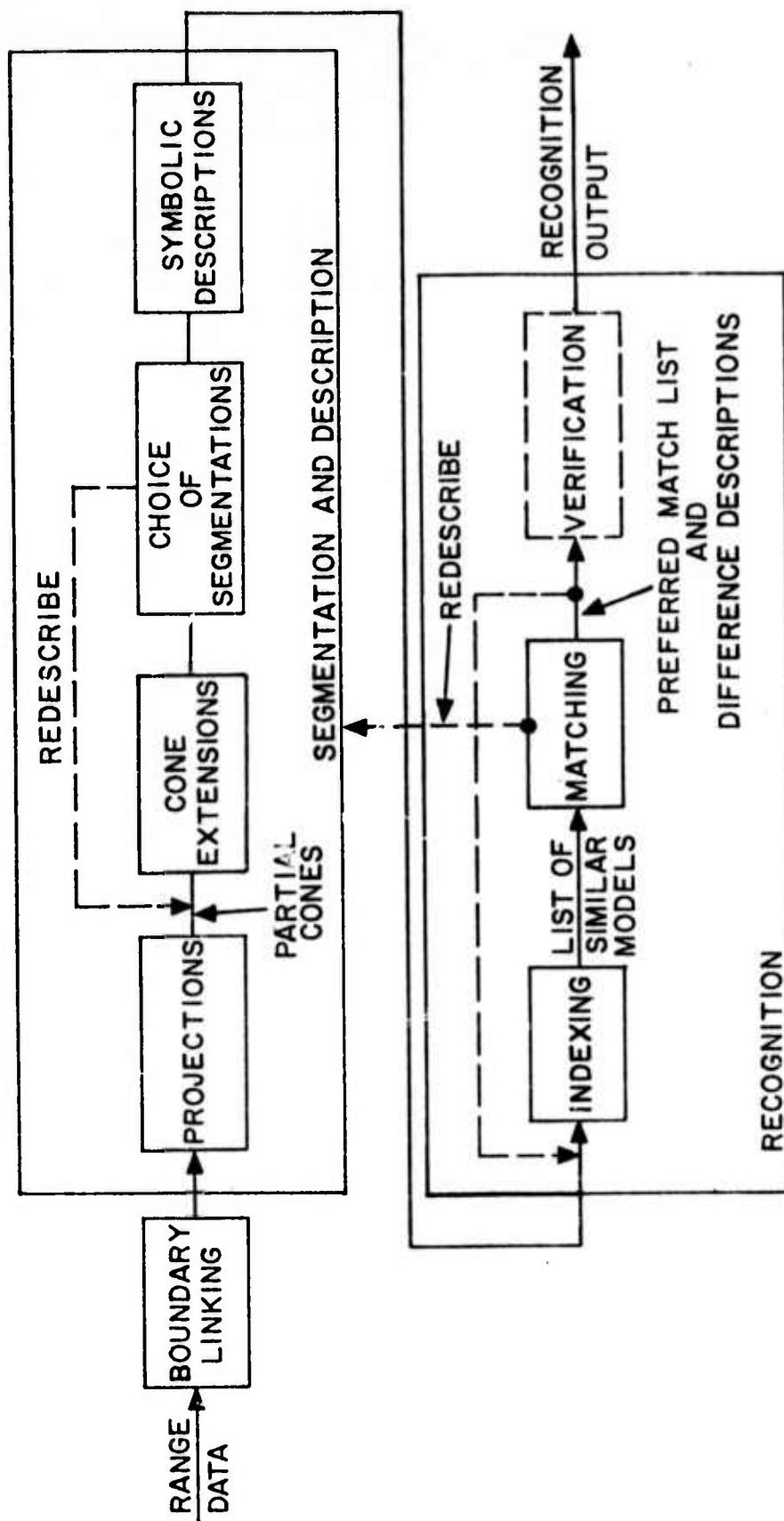


Fig. 1.1. Block Diagram showing different stages of processing

variations on the surface of the body. Boundary detection in TV images has proved to be a difficult problem, even with a restriction to polyhedral objects only. The boundaries separate different bodies in the scene; however, touching objects are not necessarily separated. This important case has been ignored. In occluded (including self-occluded) scenes, some connected parts of an object may not appear connected. (Boundary organization is discussed in chapter 3.)

Techniques for segmenting an object into sub-parts and generating the descriptions for a part as generalized cones by specifying an axis and cross-sections are not immediate from the chosen representation itself. Development of these techniques has been an important part of this work. Use of object boundaries has been important in these techniques. Our segmentation procedure starts by finding local cones, and then extends these local cones over larger areas of the object continuously, allowing the axis direction and the cross-sections to change smoothly. The extension terminates at discontinuities. Each extended cone offers the choice of a segmented sub-part of the object. This segmentation procedure often generates multiple cone descriptions for some areas of the body. Based on chosen simplicity criteria, preferred descriptions are selected from the many alternatives. The result is not necessarily a unique description. Multiple description hypothesis are generated and examined by the recognition procedures. (Details are in chapter 4.)

Symbolic descriptions of an object are generated, aiming to capture its important shape properties. They consist of the connectivity relations of the sub-parts, and summary descriptions of the sub-parts and their joints. Global descriptions depend on the relations of many sub-parts and joints, e.g. bilateral symmetry. (See chapter 5 for details).

Matching routines compare two descriptions to determine their differences. Recognition consists of choosing a previously stored description that matches best with the current description. The matching relies heavily on the structure of the object but also uses the metric properties of the sub-parts. Partial matches are necessary to recognize objects with occluded parts. Articulations of limbs are ignored; objects with different limb articulations are recognized to be the same. Efficient matching between two description structures results by the use of semantic knowledge about the descriptions, e.g. the use of *distinguished* pieces (defined in chapter 5) and the preservation of the order of the pieces at a joint (section 5.2).

The models used for recognition are not *ideal* models, we save a machine generated description of the object (any major errors are removed interactively). "Learning" techniques to generate more complete models are suggested but have not been investigated in detail.

A small number of important "features" of the symbolic descriptions are used to index into visual memory to retrieve models with similar descriptions. Indexing is necessary if the world of objects to be encountered is large in number. In that case, we cannot afford to compare the observed description with every other known description. Details of indexing, model acquisition and matching are covered in chapter 6.

(NOTE: The description and recognition chapters contain some techniques that have not been implemented in programs. These are included to provide ideas for further extensions of this work, and to indicate the possibilities of improved performance. The techniques not implemented are clearly delineated. The following chapters of this thesis are organized so that an introductory section contains the important concepts of the chapter and the details are provided in subsequent sub-sections. Appendix 2, contains a concise summary of the techniques used and has the significant program details.)

Performance:

The results of our implementation efforts have been encouraging. We are able to generate "clean" symbolic descriptions. The recognition programs can recognize objects with limbs articulated to various positions (and various viewing angles). Useful descriptions result for scenes containing multiple objects with a moderate amount of occlusion and the partially occluded objects are recognized by their partial descriptions. The methods described here are applicable to TV image processing, if suitable boundaries can be obtained.

It is our view that the important elements in judging the performance of recognition programs for the types of scenes considered here are the *classes* of scenes for which the programs work successfully. We do not have enough data for meaningful statistical results, but instead present the results of our programs on several different scenes (in chapter 7). We have used six objects for our experiments and present results on 16 different views (3 of them containing two objects). An analysis of the performance as related to the various scene characteristics is presented. We believe that these results represent a significant break from the world of polyhedral objects of the past. Section 7.2 discusses the speed and memory requirements of our programs.

More work is needed on incorporating primitives other than generalized cones in our programs for adequately describing many complex objects. We think that with the suggested additions, the programs offer potential of being useful in "real" applications to tasks such as industrial automation (particularly for "visual feedback").

Other Paradigms:

The flow of our processing of the scene proceeds in a fairly "bottom up" or *hierarchical* fashion. The necessity of a *heterarchical* control, with much interaction between different levels is widely believed to be necessary for complex visual tasks ([Winston 71]), in agreement with current psychological theories about human visual perception ([Gregory]). In the chapters on description and recognition, we indicate how such heterarchical control might be added to our programs, particular examples are those of redescription and verification. The lack of such heterarchical control in the current programs is attributed to the large effort that had to be spent in the construction of the current description and matching routines. The performance of the current programs is just adequate to distinguish between a doll and a toy horse. We believe, that addition of verification and goal-directed low level description of such features as termination of parts will greatly add to the power of the system.

More recently, Freuder ([Freuder 73a,73b]) has argued for the necessity of the intimate use of goal directed knowledge at all levels of description, in contrast to the paradigm of generating descriptions and matching them to models. The author feels that this is desirable; however, a principal problem to be overcome is the selection of the model to guide the descriptions. In special restricted applications, such as looking for a specific object, this knowledge may be easily available. In a more general situation, however, we believe that descriptions of the complexity described here need to be generated before a likely model can be retrieved from the memory. Local descriptions can potentially match a very large number of objects and are unlikely to be useful in guiding further descriptions.

The techniques presented here may be considered as modules that would be useful for

addition of other primitives or be adapted for specific applications using a different control structure. These modules should also be of direct use for extension to more complex scenes, such as heavily occluded scenes.

CHAPTER 2

REPRESENTATION

We are interested in the description of the "shape" of an object, and in the recognition of an object based on its shape description. The term "Shape" has intuitive meaning for us, but it defies a precise definition. The dictionary equivalents of form or extent are equally imprecise. We are then interested in descriptions that capture our intuitive notions of what shape descriptions should be like. An array of positions of points on the surface is a complete description of the object and useful for some purposes, but it hardly describes what one generally thinks of as shape.

Among the desirable attributes for a shape representation are: the representation should describe a set of shapes compactly and simply, and should allow for determination of similarities as well as differences between two shapes. Incremental changes in an object should reflect as incremental changes in the description. Many "universal" representations have been proposed, e.g. expansions in orthogonal series such as moments or Fourier series, or descriptions of surfaces by two-dimensional splines. These representations contain no sense of segmentation into parts. Local, incremental change of shape does not result in a local or incremental change in its expansion in an orthogonal series. It is unlikely that a single representation will be suitable for describing all shapes; we present a representation that describes a certain, hopefully wide and useful, class of shapes simply and compactly.

It seems to us, that any intuitively appealing shape description must represent complex objects by segmentation into simpler sub-parts. The segmentation criteria could be simplicity of sub-parts (is a function of what a simple primitive is), articulation characteristics (each moving volume is a separate part) or be based on our knowledge of the construction of the object (such as knowledge about certain parts having been attached to others). This segmentation and the connectivity relations of the sub-parts comprise the "Structure" of the object, hence our use of the term "Structured Descriptions". Segmentation allows for incremental changes of object to be described incrementally.

Primitives may be surface descriptions or volume descriptions (for the simpler case of polyhedral objects, edge descriptions suffice). For three-dimensional objects, the volume primitives provide more intuitive segmentations. Surface discontinuities are usually not a good basis for segmentation. For some objects a particular surface is of special importance, e.g. many parts might attach along a flat surface. In such cases the representation should use a combination of surface and volume descriptions.

We use Generalized Cones as main primitives; other primitives are allowed. The representation chosen has been previously described in [Binford] and [Agin 72]. Here, in sec. 2.1, we present only a brief summary, reflecting our interpretation of it, and to allow an independent reading of this thesis. Symbolic descriptions of these parts, their joints and the complete object are discussed in chapter 5.

2.1 GENERALIZED CONE PRIMITIVES

An object is represented by segmenting it into sub-parts. Different parts attach at a *joint*. A sub-part may have its own sub-parts, depending on the amount of detail to be

represented. This provides a hierarchical representation allowing for varying amount of detail to be stored. A decomposition of a human shape is as shown in Fig. 2.1. The human shape is represented as being composed of two legs attached to one end of the body and the two arms and the head attached to the other end of the body. Arms can be further represented as consisting of upperarm, forearm, and the fingers of the hand.

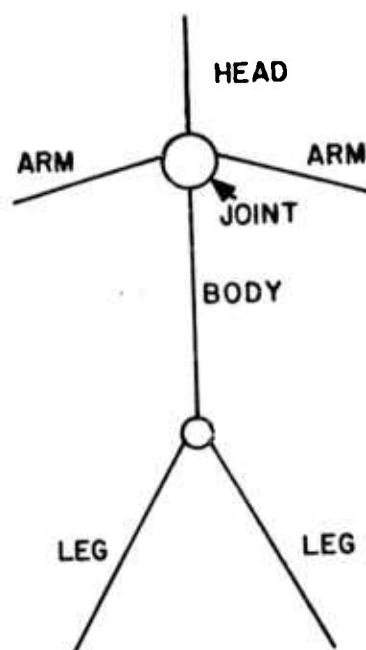


Fig. 2.1 Segmentation of a Human Form into Sub-parts

The principal representation for the primitive parts in our system is by generalized cones; other primitives are allowed. A generalized cone is defined by a space curve, called the *axis*, and normal *cross-sections* along this axis. The cross-sections may be any planar area, and the cross-section shape may change along the axis; the function describing these cross-sections is called the *cross-section function*. If the cross-sections do not change along the axis then the generated volume is a generalized *cylinder*. Formally, the volume described by sweeping of the cross-section along the axis has been formulated as *Generalized Translational Invariance* by Binford ([Binford]). We impose the following constraints on the axis and the cross-sections:

- 1) The cross-sections must be normal to the local axis.
- 2) The axis must pass through "corresponding" points of the cross-sections.

The points of the cross-sections to be used as corresponding points need to be chosen. Intuitively, we want these points to be the "centers" of the cross-sections. The centers of gravity seem to be appropriate and are taken to be the ideal choice for the corresponding points (note this choice is being made as a matter of definition). The choice of corresponding points may follow from additional constraints on the generalized translational invariance. The centers of gravity require the knowledge of complete cross-sections for their computation. In section 4.1 we present another choice of corresponding points that are more directly computed and approximate the centers of gravity.

The constraints stated above do not necessarily determine a unique axis, cross-section description for a given volume. E.g. a rectangular solid could be described by axes parallel to any of its three sides. However, for an untruncated straight, circular cylinder these conditions do determine a unique description, corresponding to the usual choice of axis. Some axis, cross-section descriptions are preferred to others, as discussed further in sec. 4.5. The problem of locating axis without access to the complete cross-sections will be discussed in sec. 4.1.

Note that this representation has not specified an algorithm for segmentation of an object into sub-parts. Each segmented primitive is to be a simple and continuous generalized cone; the conditions for determining simplicity and continuity will be further established in sec. 4.3. Even with specified continuity conditions, segmentation of an object is not straight forward and our technique is described in chapter 4.

The choice of generalized cones as primitives is attractive for describing shapes with an axis along which the cross-section varies smoothly. This is often true of elongated shapes (but not restricted to them). Elongated shapes are commonly found in both man-made and natural objects, e.g. limbs of animals, machine shafts, legs of a table, handle of a hammer etc., and a large class of objects can be conveniently described as being built of generalized cone parts. A program dealing with a wider class of objects will also need additional primitives, such as planes, spheres, and surfaces.

The shape of a primitive consists of the shapes of its axis, the shapes of the cross-sections along this axis. A cross-section can be described by techniques of segmentation into primitive two dimensional "cones", i.e. the same representation methods can be scaled down from three dimensions to two dimensions. The shape descriptions of the axis and the cross-section function are problems of description in one dimension. Again, segmentation into primitives, perhaps linear or "continuous" segments, suggests itself. The detail of these shapes in the representation can vary with the use that they are put to; we have not concentrated on these details here. We have mainly been interested in the structure of an object and use only crude descriptors to represent the shape of the individual primitives. These descriptors are discussed in Sec. 5.1.

Objects with holes can be described in terms of the solid matter that they are made of, but descriptions in terms of holes are simpler and carry more semantic information. The holes are viewed as negative volumes, and can be described as negative generalized cones (or as one of the other primitives listed in sec. 2.2).

The chosen axis, cross-section representation has similarities with the Blum medial axis transform ([Blum]). The main differences are as follows. The Blum transform is sensitive to small changes in the boundary or the surface (a small disturbance causes major excursions of the axis) whereas for the generalized cone a small disturbance merely perturbs the local cross-sections. Computation of the Blum transform requires knowledge of complete surface, our method is content to compute the partial cross-sections. The Blum transform is a "transform", i.e. it yields a unique representation for given data, whereas multiple cones can describe the same volume effectively (e.g. a rectangular solid may be represented by axes in any of the three orthogonal edge directions). Non-uniqueness of the representation is not viewed as a disadvantage, but rather an important advantage allowing for alternative descriptions. The Blum transform is well defined, however, while the description mechanisms described here are still evolving. A more detailed comparison may be found in [Agin 72].

2.2 OTHER PRIMITIVES

Some shapes need a complex cross-section function when described as generalized cones. Description in terms of other primitives may be simpler. In the following we suggest additional primitives. This list is only meant to *widen* the class of shapes that can be well described.

Spheres: Though spheres can be represented by an axis, cross-section representation, they do not have a preferred direction of elongation, and description as a sphere is simpler. Parts of spheres can be described as terminated spheres.

Surfaces: We have argued for the desirability of volume representations. However, for some objects, a particular surface has special meaning and description in terms of this surface is preferred. E.g. the top of a table may be described as a thin cylinder or as a flat surface. Surface descriptions are likely to be useful for objects made of thin material, such as folded sheet metal. Surfaces are also useful in describing terminations of cones.

Terminations: A cylinder (cone) terminated by a surface not normal to its axis can be described as a cone with a tapering cross-section function near the termination. However, a much simpler description is as a cylinder (cone) and a terminating surface.

The programs we present use generalized cone primitives exclusively. Future incorporation of other primitives is compatible with the methods used. These primitives suffice for many shapes, e.g. toy animals, hand tools, and some machine parts (shafts). A major class of objects that is hard to describe by primitives discussed here is that of complicated castings; perhaps there are no simple representations for such shapes. We have not implemented important surface descriptions, but think that with the addition of such a primitive, useful descriptions can be generated for a large number of objects encountered in applications such as industrial automation.

CHAPTER 3

DATA ACQUISITION AND BOUNDARY ORGANIZATION

In this chapter we describe the technique used to directly measure three dimensional positions of points on the surface of an object and inference of object boundaries from this data. Also discussed is the separation of a scene into different bodies, using the derived boundaries.

Humans are able to view photographs easily and infer depth information from a single picture using many cues, such as texture gradients, shadows, highlights etc. However, machine implementations of these depth inference techniques constitute significant research problems by themselves. Our decision to use direct depth ranging was as an expedient, so that we could investigate the problems of shape descriptions. Note, that we do not have "complete" information about an object, only the positions of points on the visible surface. Most of the "perception" problems thus remain. It has turned out that many of the techniques developed can be applied to TV image data, and even provide clues for attacking this problem.

In sec. 3.1, we briefly describe a laser triangulation ranging method, originally developed by G.J. Agin and T.O. Binford. The geometry of the current setup is different from that described in [Agin 72], however the description of details is still applicable. A reader familiar with Agin's ranging method may skip sec. 3.1. A similar ranging method has also been described in [Shirai]. Some other methods of depth ranging are discussed in [Earnest].

3.1 LASER TRIANGULATION RANGING

Ranging by laser triangulation is similar, in principle, to ranging using a stereo pair of pictures, with one camera replaced by a known source of light. Consider an object illuminated by a single light beam of known position and orientation (Fig. 3.1). The camera image consists of just the one illuminated point. If the camera is calibrated ([Sobel]), the ray from the image to the object point is known. Since the illuminating beam is also known, the position of the object point can be directly determined by triangulation. Position information for the whole object can be obtained by scanning the object by a number of known rays. However such a scan requires a large number of beam positions and would be slow.

Consider the illuminating light beam to be replaced with a plane of light, of known position and orientation. The plane intersects the object along a planar curve, and this curve forms an image on the camera screen. With each point on this image, we can associate a ray to the object, as before. Now, the intersection of this ray with the light plane uniquely determines the position of the object point. Thus we can determine the 3-d position of each point in the image of the illuminated part of the object. The scanning of the complete object now involves sweeping known planes across it, which is significantly faster than scanning with a point beam.

The apparatus used for generating scanning light planes is shown schematically in Fig. 3.2. Light from a laser is diverged to a plane beam by a cylindrical lens. The diverged beam is reflected by a mirror which can be rotated about an axis, to generate different output planes. These planes all pass through a common line, but near the object they may be considered nearly parallel, but displaced in position. The camera looking at the object sees only the laser light, either

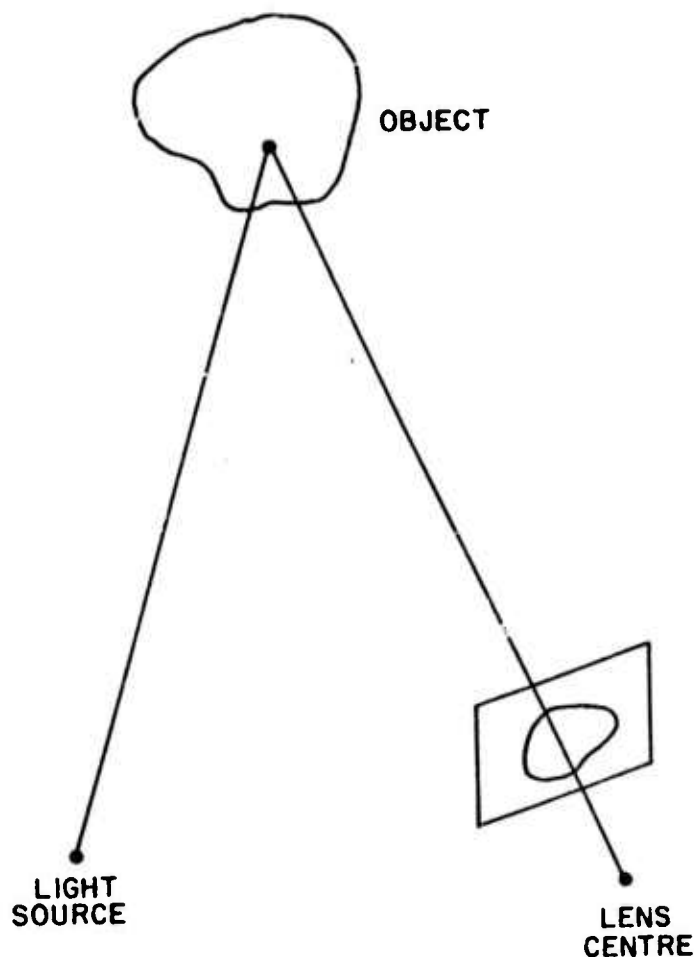


Fig. 3.1 Schematic View of Triangulation Ranging

by proper contrast adjustment or by placing an interference filter in front of the lens. The light plane is scanned across the object and the corresponding images on the camera screen are recorded. The plane positions are known by a calibration procedure and three-dimensional positions of the points on the image can be computed.

Surfaces that are parallel to the light plane, are measured with poor accuracy. To counter this, we choose another orientation of the light plane, obtained by rotating the cylindrical lens in the path of the laser beam, and sweep the object with planes of this new orientation (by rotating the mirror). The optimum angle between the two orientations is 90 degrees, however hardware limitations of our apparatus frequently limit the allowed angle to about 45-60 degrees.

Our data input thus consists of two series of scans, each series of scans consists of nearly parallel but displaced light planes, and the two orientations are at an angle of between 45-90 degrees. Figs. 3.3 shows the two series of scans for a doll. Each frame of a scan consists of the set of points in the camera image that have non-zero brightness. With each frame is associated a transform matrix. Given an image point in the frame, this matrix can be used to generate the three dimensional position of the corresponding object point (use of homogeneous coordinates ([Roberts 65]) allows the transformation to be a simple matrix multiplication operation, see [Agin 72] for details).

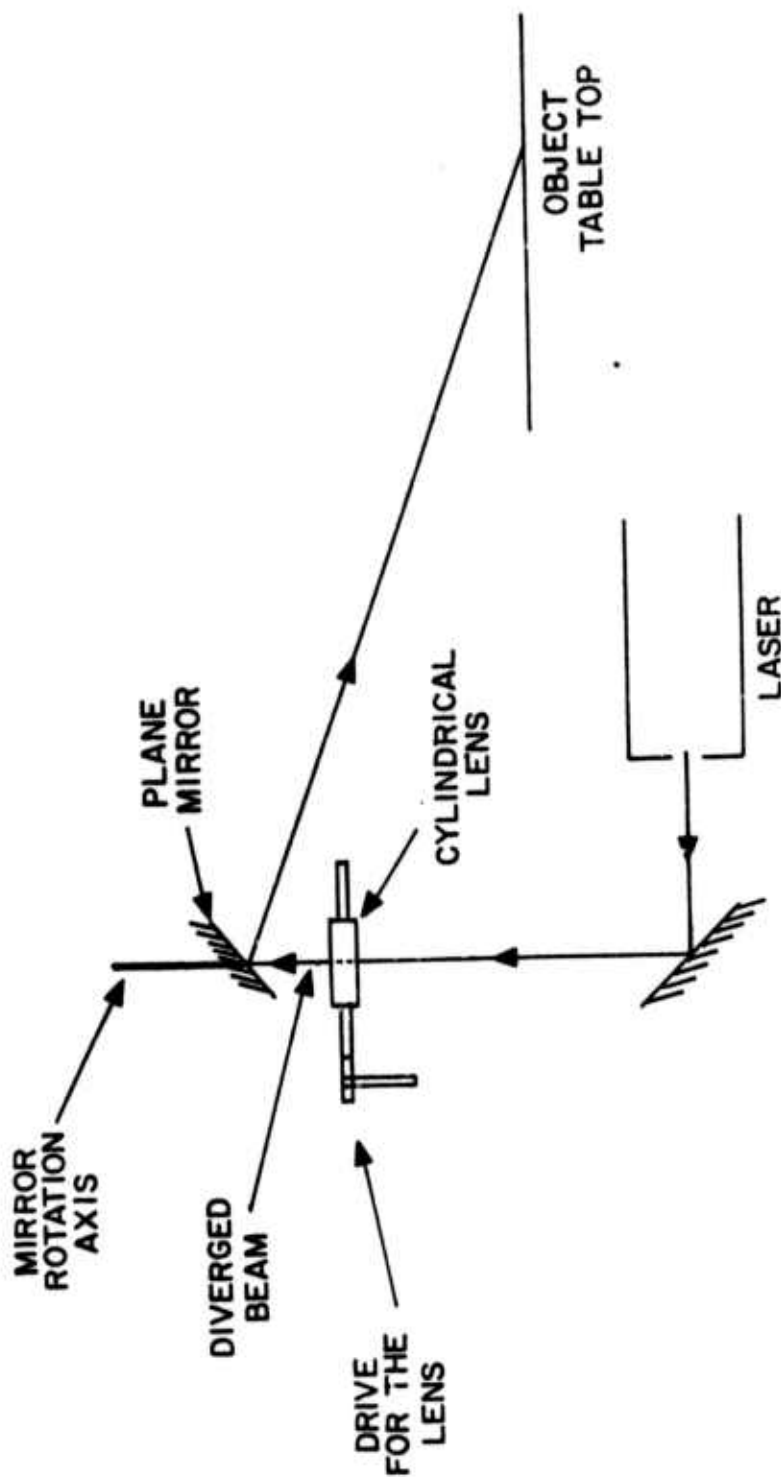


Fig. 3.2. Schematic of the ranging apparatus

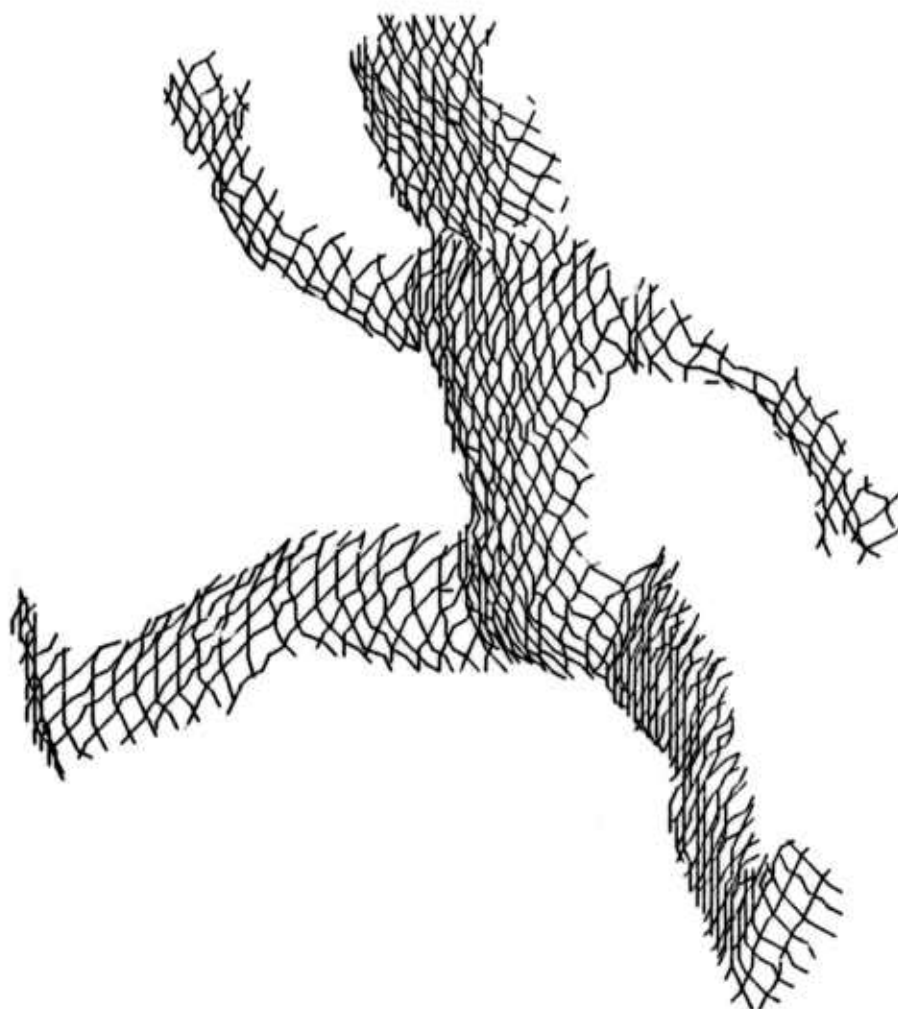


Fig. 3.3 Laser Scans for a Doll

The triangulation angle, i.e. the angle between the light source and the camera at the object is typically about 45 degrees. For such a setup, the resolution of the ranging system (relative error) is about 1 mm at a distance of 1 meter. This system is subject to occlusion from two angles; the observed surface of the object must be visible from the camera view-point, and also not be in a shadow from the laser point of view. Thus for a circular cross-section, we are able to see only about 120 degrees of the cross-section. We have a trade-off of shadows for accuracy in deciding on a triangulation angle.

The speed of the data acquisition is intrinsically limited by the time required to read the TV camera for each plane position. We have not attempted to minimize the data acquisition time and the scanning of an object typically takes a few minutes. Applicability of "grid coding" schemes to speed up the ranging process is discussed later, in sec. 7.2.

This method of depth ranging is attractive because of the direct measurement of range. The author was experimenting with stereo measurement of depth at the beginning of this research ([Nevatia]). The problem of finding corresponding regions in two scenes is a time consuming and error-prone operation there, and the author was easily converted to using this ranging method so that work could concentrate on the problems of shape description. (The description techniques to be described are equally applicable to range data obtained by other means.) Baumgart ([Baumgart]) describes some techniques for data acquisition using multiple TV images. Other relevant work on stereo depth measurement may be found in [Hannah] and [Levine].

The present implementation with a He-Ne laser, limits the hue of the objects whose range can be measured. Use of a bright white light source or a multi-colored laser source would alleviate this problem. The main disadvantage of the method is in the shadows caused by wide angle triangulation (a much smaller triangulation angle would still be useful). Range of the apparatus is limited by the power required to project a plane, even with relatively efficient imaging devices, such as silicon target multiplying tubes.

3.2 BOUNDARY ORGANIZATION

The data from the laser scans of the scene consist of two series of scans. Each scan consists of several *frames*. Each frame is composed of the points of non-zero brightness in a single TV image (corresponding to one position of the illuminating light plane). These points correspond to the parts of the object illuminated in that particular frame. The three dimensional positions of these points are computable by use of the known calibration information.

Each frame contains a number of *connected segments*, corresponding to continuous surfaces of the objects scanned. A discontinuity in the object surface appears as a discontinuity in the camera image of the laser scan. The space discontinuities also correspond to the object boundaries (as viewed from the particular angle). Thus the outer boundaries of an object can be constructed from the extremities of the connected segments in the laser scans. The notion of a boundary as defining the extremes of the continuous surface, agrees with the normal concept of a boundary (as opposed to texture or color boundaries for example). However, in some instances of touching objects this process will result in boundaries which include parts of more than one object.

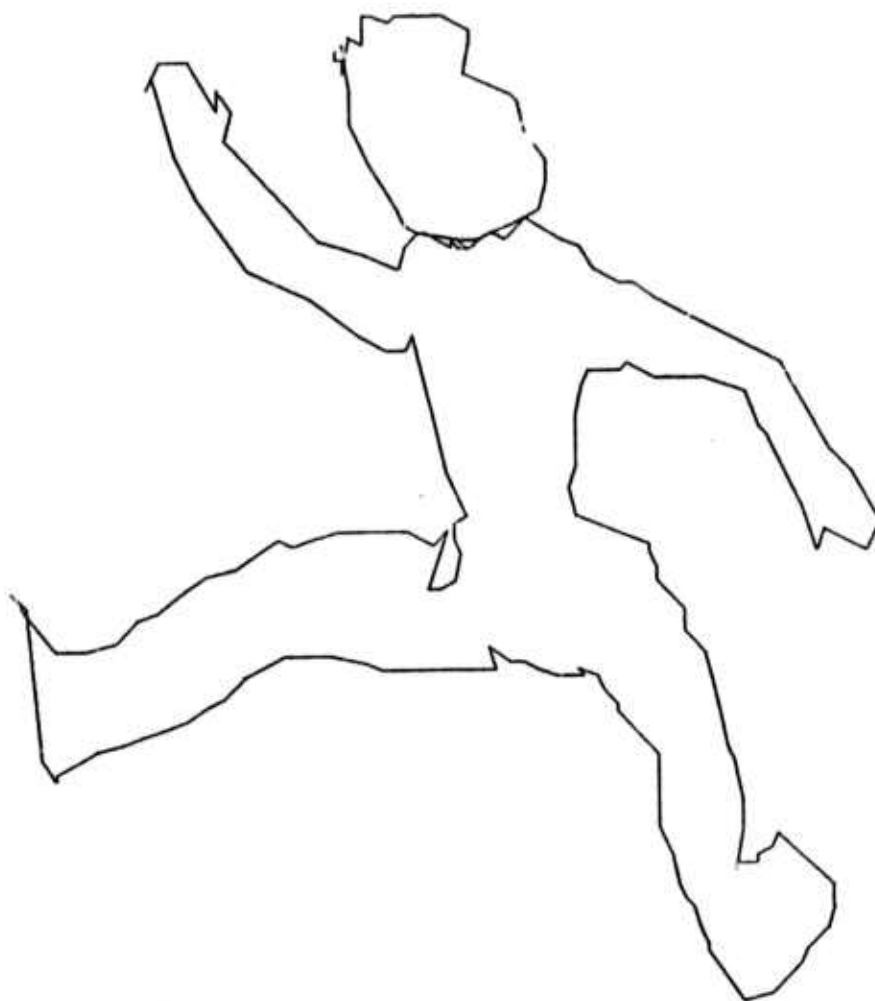


Fig. 3.4 Boundary Constructed from the Scans in Fig. 3.3

The processing of the input data requires locating the connected segments in a frame, and locating their extremities. Ideally, the points in a segment would connect to form a thin curve. However, due to several factors including the finite width of the illuminating plane and vidicon blooming, these points form an area, several raster units wide. A "thin" curve approximation to this area is obtained (all laser scan pictures presented in this thesis display thinned scans). Some thinning techniques were presented in [Agin 72]. Alan Borning has implemented improved thinning techniques and they have been used for pictures here. These thinning techniques are not of direct interest here and no further details are provided.

The extremities of such segments are linked (by straight lines) to form a complete outer boundary for the objects in the scene. The details of the algorithm for constructing such boundaries and also the likely sources of errors have been relegated to Appendix 1. The reader may assume the laser scans and a boundary to be the input for the algorithms described in the succeeding chapters. An example of the boundary output is shown in Fig. 3.4 (from the laser scans of Fig. 3.3).

The construction of such a boundary provides a useful and convenient way of structuring the data. Body separation and detection of holes follow immediately from the boundary data (details of body separation are discussed in sec. 3.3). The boundary is believed to be of importance for human visual perception ([Attneave]). The description routines presented in the succeeding chapters rely heavily on the use of such a boundary, and this information alone is sufficient for many applications including recognition of many scenes. The performance improvements of our description routines over previous work ([Agin 72]) are strongly dependent on our use of the boundary data (see Chapter 4). (Note that we do not generate descriptions of the boundary *per se*, rather descriptions of the volume outlined by the boundary.)

3.3 BODY SEPARATION

Separation of multiple objects in a scene from the object boundaries is direct. These boundaries correspond to depth discontinuities in object space. Each isolated set of boundaries defines a body that is connected in space. This set contains more than one boundary if the body has holes. However, parts of a connected body may not always seem connected, because of shadows or occlusion. We have a partial body separation; a body may be split in more than one piece, but all separate bodies have been isolated. However, bodies which touch are not necessarily segmented. E.g. consider the TV image in fig. 3.5, the laser scans for this scene are shown in fig. 3.6 and the boundary output in Fig. 3.7 (more examples are presented in chapter 7). The separation of the left doll leg from the snake is difficult in the TV image, but the separation of this snake from the upper part of the leg is immediate from the boundary data. Note that the lower part of the leg is seen as connected to the snake, as the two objects touch and no depth discontinuity is observed. (If the lower leg were not connected to the snake, it would still appear separated from the rest of the doll.) More sophisticated segmentation techniques will be required for separating touching objects. The problem is related to that in inferring body segmentation in monocular scenes and has not been investigated here.



Fig. 3.5 A TV Picture of a Doll and a Snake

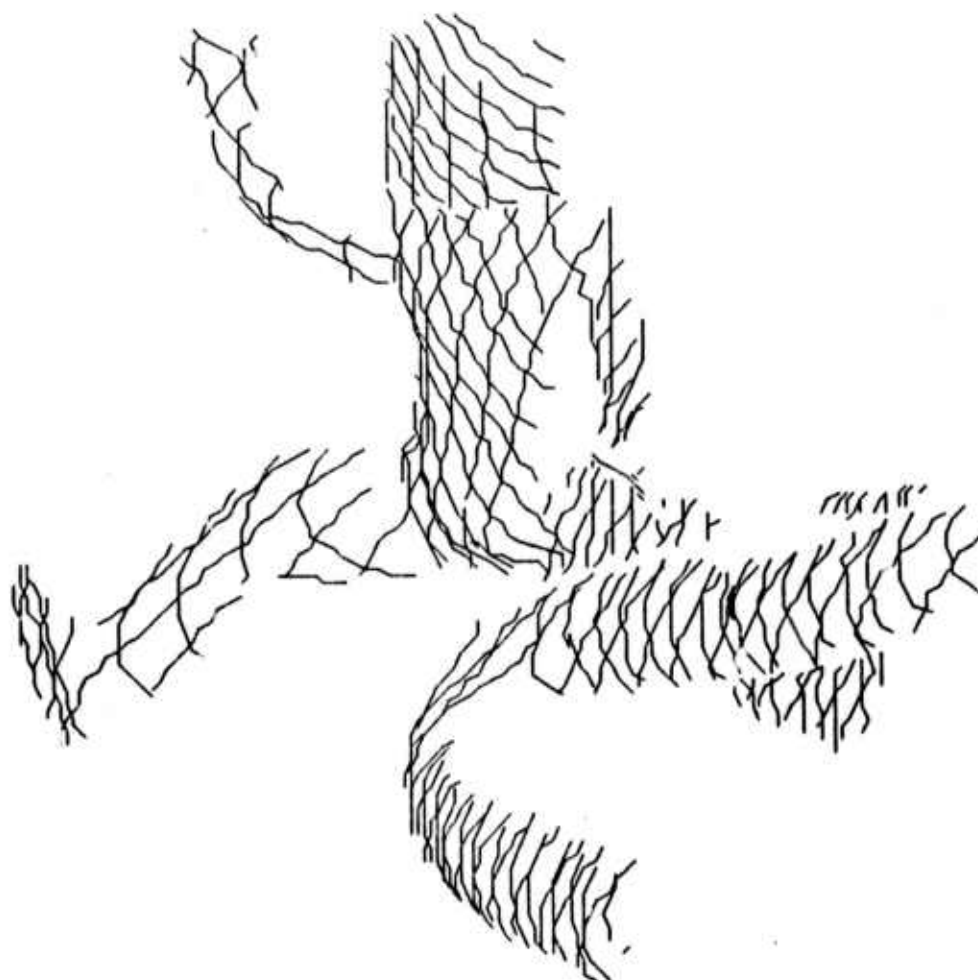


Fig. 3.6 Laser Scans for the Scene of Fig. 3.5



Fig. 3.7 Boundary Derived from the Scans of Fig. 3.6

CHAPTER 4

BODY SEGMENTATION AND PRIMITIVE DESCRIPTIONS

Our description scheme is based on describing an object in terms of simpler sub-parts. Generalized cones are used as primary primitives; other primitives such as planes and spheres are necessary, but have not yet been implemented. Generalized cones can describe arbitrarily complex shapes. Simplicity criteria need to be specified to permit their use in segmentation. We segment an object into generalized cones with a "smooth" axis and cross-section function, i.e. the axis direction and the cross-sections along the axis change continuously. Continuity is a natural basis for segmentation, but it is clear that the resulting segmentation into primitives will depend strongly on the specification of the continuity conditions. We do not expect a *perfect* segmentation for every object, in the sense in which humans would segment it. Context must be used to join some segmented parts or further segment a part at some higher level. Alternate descriptions are used when multiple description hypotheses are reasonable. (The recognition programs examine the multiple hypotheses and select the one that matches best.)

In this chapter we discuss the techniques that generate a number of alternate segmentations and the basis for choosing among the alternatives. The following chapter covers further symbolic descriptions for the selected segmentations. The body separation was discussed in sec. 3.3; in this chapter, we will be concerned with descriptions of one body.

The chosen representations do not provide a direct computational procedure for generating segmented descriptions from the input data, unlike *transform* representations, e.g. the Blum transform or the Fourier Transform. (Local descriptions can be directly computed in our representation by fitting cones to the local data.) Continuity and simplicity conditions are usable for examining the acceptability of a cone description. However, no a priori knowledge of the axes directions, axes shapes, the cross-section sizes or the cross-section functions is available.

Our segmentation technique proceeds in two parts. First, the areas of the body that can be described by local cones are determined by the use of the "projection" technique (discussed in Sec. 4.2). The second part improves on the axes of the local cones determined by projections and then extends these local cones, by allowing the axes directions to change smoothly (as discussed in Sec. 4.3). Such extensions allow tracing of slowly curving cones. The extensions terminate if the cones cannot be extended continuously, either having reached the end of the object, a cross-section discontinuity or an axis discontinuity. Other cone description methods are discussed in section 4.4.

A number of local cones are generated and then extended. Each extended cone represents a possible segmented sub-part. Many local cones are likely to extend to common parts of a body. Thus a number of alternate segmentations are suggested. We choose among the suggested descriptions and retain a small set of alternate descriptions. The result is not necessarily a unique description for an object, but neither do we wish to retain all possible combinations. Simple preference criteria select preferred descriptions. Among two descriptions for the same area, we prefer a long cone to a short cone; and prefer cylinders to cones. Descriptions of areas contained in areas described by other cones are eliminated. When a clear choice is not available, alternate descriptions are made. The choice of segmentations is discussed in section 4.5. The selection procedures used are local. Larger context, e.g. the context of a joint for choice of local descriptions has not been investigated. This has been satisfactory for scenes of moderate

complexity; more global choice is clearly useful. In section 4.5, we also discuss techniques of redescription of parts with more context, likely to be useful for improved descriptions.

Our description procedures use only the boundary of the object. The 3-d data has been utilized in constructing the boundary from the depth discontinuities. We also use the 3-d position of the points on the boundary. We compute only those points of a cross-section that lie on this boundary, and no assumptions are made about the shape of the cross-section a priori. The remainder of the cross-section can be computed on demand. (See sec. 4.1 for locating the axis of a cone from only partial knowledge of the cross-sections.) The details of the cross-sections have not been useful, because of the limitations on the visible part of the cross-sections and the errors of ranging (see sec. 4.3). Also, we feel that the details of the interior are of secondary importance, useful for making finer distinctions. This is in agreement with psychological evidence about human perception; crude boundary information is enough for many recognition tasks ([Attneave]). The boundary does depend on the viewing angle, but the results produced are relatively insensitive to the viewing angle over a wide range. Note that we do not make descriptions of the boundaries themselves (viewed as space curves), rather of the *volume* outlined by the boundaries. Use of the boundary permits us to use the same techniques of analysis for processing data from TV images only. The boundary must now be obtained from intensity information. However, boundaries from intensity information are difficult to obtain and unreliable. The problem of body separation must also be solved by other means. (This problem is similar to the problem of separating touching objects.)

In previous work ([Agin 72]), Agin has described procedures to generate cone descriptions. However, major shortcomings of these techniques limit their performance on moderately complex scenes, making them unusable for further extensions. His methods fit cylinders of circular cross-sections to the visible surface of the object. These methods had no well defined notion of a part, and a cylinder would often include two proximate but distinct parts of an object, such as two fingers of a glove. Such errors cannot be easily corrected at a higher level by use of context. No attempt was made to connect the separate cones to form an object in Agin's work. Our description process is more structured because of its use of boundary. Our techniques are conceptually different and their development has required a large investment of effort. They exhibit substantially improved performance; some examples are presented in sec. 4.3. Our programs are also substantially faster, as we need to work only with the boundary of an object. Also, our methods do not assume any particular cross-section shapes, whereas Agin's methods were restricted to circular shapes.

Each cone description is represented by a list of axis points and normal cross-sections along this axis. Summary descriptions for each cone include the length of its axis, the average width of the cross-sections and the ratio of the length to the width. The cross-section function is approximated by a linear function and an average cone-angle is computed. These summary descriptions are discussed in more detail in section 5.1.

4.1 CONE DEFINITIONS

The generalized cone representation has been discussed earlier (sec. 2.1). The constraints on an axis, cross-section description were defined to be that: the cross-sections must be normal to the local axis and that the axis must pass through corresponding points of the cross-sections. Choice of centers of gravity for corresponding points was considered.

However, the center of gravity of a cross-section cannot be computed without the knowledge of the complete cross-section. We see only the front of the cross-section. If cross-section shapes were limited, we could estimate the complete cross-section by fitting these various shapes. An alternative approximation would be to use the center of gravity of the visible cross-section. In this implementation, we compute only the two end-points of a cross-section (those on the boundary but with known three-dimensional positions) and use the mid-point of the line joining these two end-points. This method of determining corresponding points gives a closer approximation to the center of gravity for the case of circular cross-sections. Fig. 4.1(a) shows the axis obtained by joining the centers of gravity of the visible parts of the cross-sections and Fig. 4.1(b) shows the axes obtained from the mid-points of the ends of the visible part of the cross-sections. The axis in Fig. 4.1(b) coincides with the desired central axis. However, in our system less than half the cross-section is visible and the approximation is not perfect.

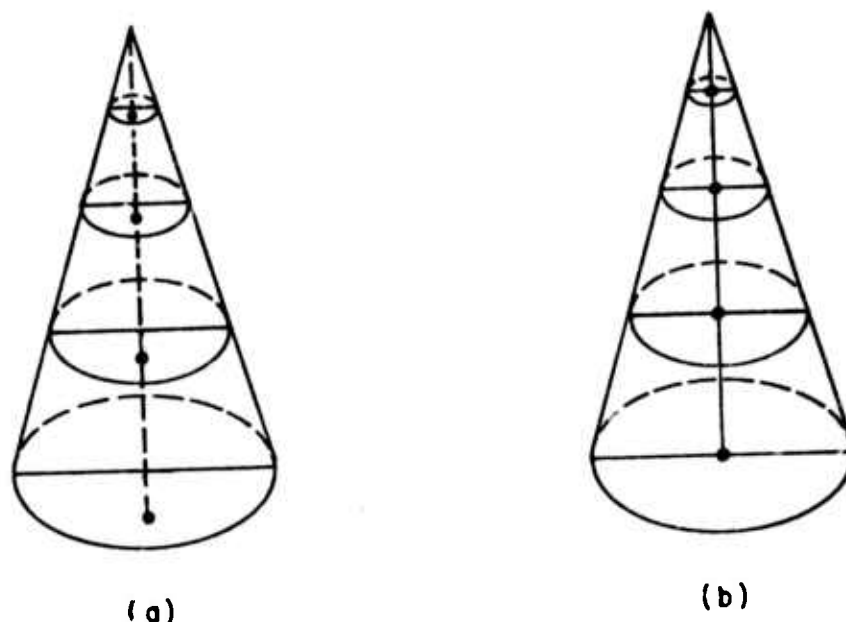


Fig. 4.1 Two choices for Axis Points

For computing cone descriptions, we have taken the mid-point of the ends of the visible part of the cross-sections as our choice of the corresponding points. This is taken to be the *definition* for axis, cross-section descriptions, (i.e. we require the axis to pass through these points). Note that this choice of corresponding points will cause a cone axis to be located in somewhat different positions with varying viewing angles, though the variations will be small for elongated parts. Our recognition programs do not rely on the precise location of such axes and are insensitive to such variations.

4.2 LOCAL CONE SAMPLES: METHOD OF PROJECTIONS

As the first step in finding cones describing an object, we find local cones describing small areas of the object. If two consecutive parallel cross-sections have the property that their

mid-points join in a line which is normal to these cross-sections, then this line and the cross-sections comprise a local cone by our definitions of a generalized cone. We find local cones satisfying these conditions, by constructing cross-sections normal to eight equally spaced directions, by using "projections" as described below.

Consider a particular projection direction, say X_i , having a specified orientation with respect to the object. We wish to find local cones with axis pointing along this direction. Rotate the image (about the origin) so that X_i coincides with the unrotated X-axis. Fig. 4.2 shows the doll of fig. 3.4 so rotated by 45 degrees (X_i is pointing horizontally). Now construct cross-sections normal to the rotated X_i , spaced 10 raster units apart (the complete picture is 330 units wide), by forming pairs of points on the opposite sides of the boundary. As example see (p1,p2) and (p3,p4) in Fig 4.2. Note some cross-sections in this figure are not exactly vertical; this is because of coarse sampling of the boundary and lack of interpolation between boundary points. If two consecutive cross-sections satisfy the condition that the line through their mid-points is within a specified angle (22.5 degrees) of X_i , we have found a local cone (actually an approximation to one). One local cone may contain more than two cross-sections, if other consecutive cross-sections satisfy the constraints in successive pairs. Fig. 4.3, shows the axes obtained from the cross-sections of Fig. 4.2. (The axes are shown by double lines and the associated boundaries are shown in heavy lines). These are the parts of the object that have local cone descriptions with the axis pointing in the chosen projection direction. Fig. 4.4, shows all local axes obtained from projection in eight different directions for this object (each 22.5 degrees apart). More program details are described in Appendix 2.

The parameters used for this method were determined empirically. The accuracy with which the axis can be determined (within 22.5 degrees of the projection direction in the above description) is dependent on the spacing between two neighboring cross-sections and the expected random variations in the boundary. Also, if the axis needs to be determined more accurately, we need to project in more directions. However, the techniques described in the next section for refining the axes directions are more efficient. Four projection directions are usually adequate for finding all local cones of interest (with the chosen accuracy range); eight directions provide enough redundancy. The choice of spacing of the cross-sections along the projection direction is by a trade-off between the spatial resolution with which the local cones can be determined and the accuracy of the axis direction.

The resulting segmentation for an object is directly dependent on the local cones generated by projections. The projection methods are successful in finding local cones for elongated parts of an object. Local cone generation for non-elongated parts (with length to width ratio of less than 0.5) is not reliable.

4.3 EXTENSION OF LOCAL CONES

The projections provide us with a number of local cones and their approximate axes directions. In this section we describe a procedure to improve the axes directions and extend the cones as far as possible continuously (a concise description of the algorithms and more implementation details are in Appendix 2). Extensions of a cone allow the axis direction to change smoothly.

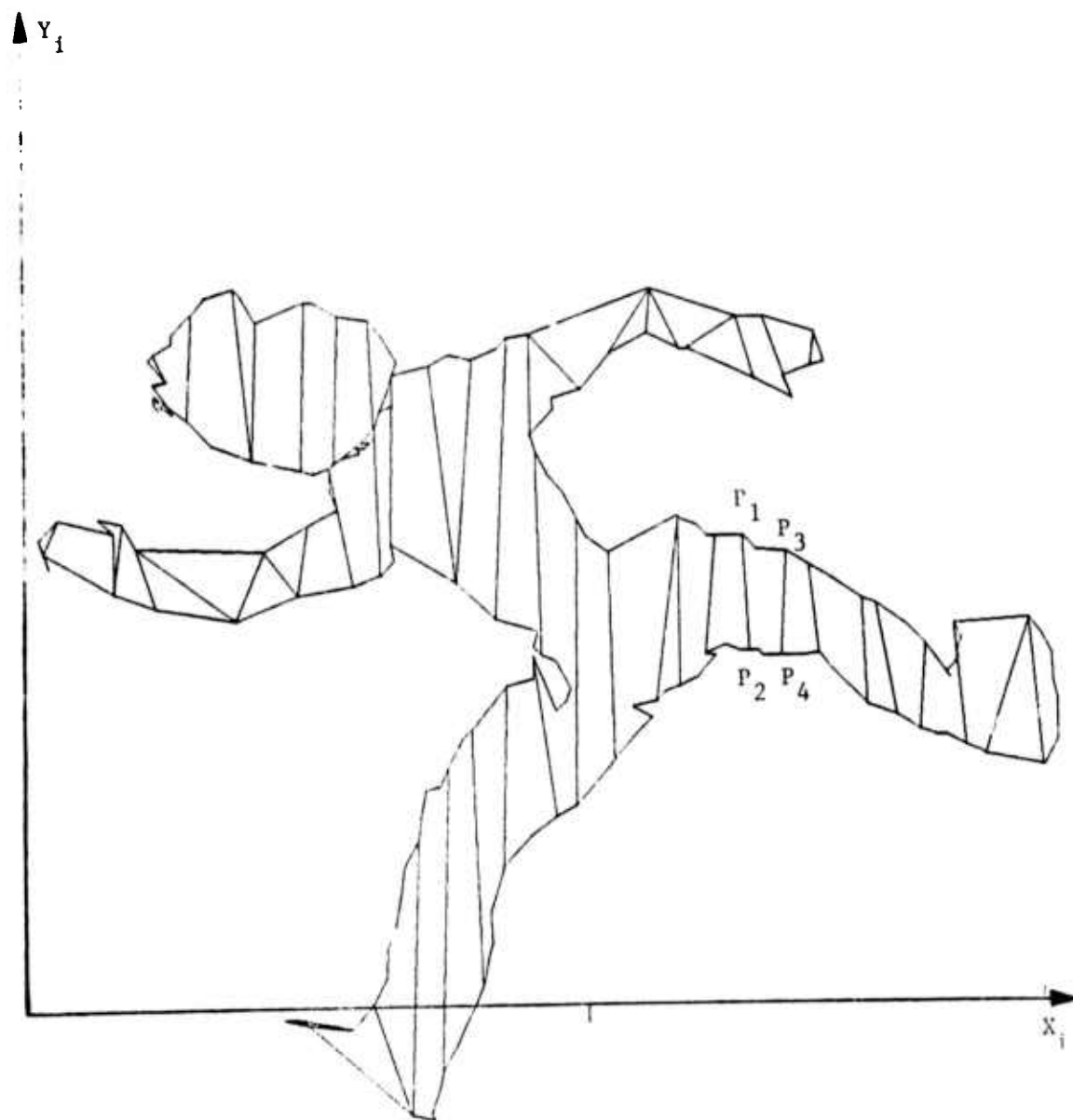


Fig. 4.2 Cross-sections from Projection in a Chosen Direction

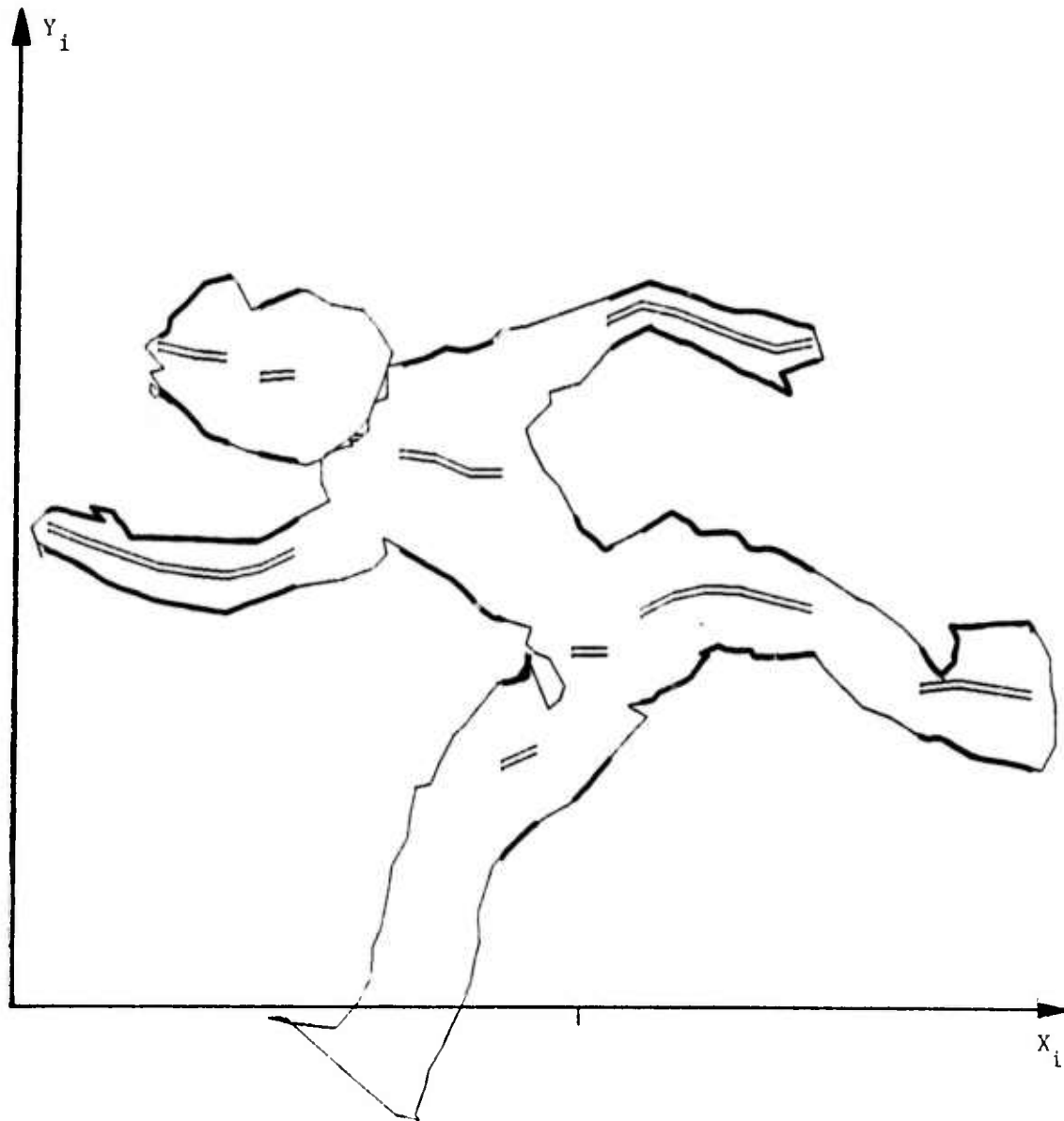


Fig. 4.3 Local Cones Generated from Fig. 4.2

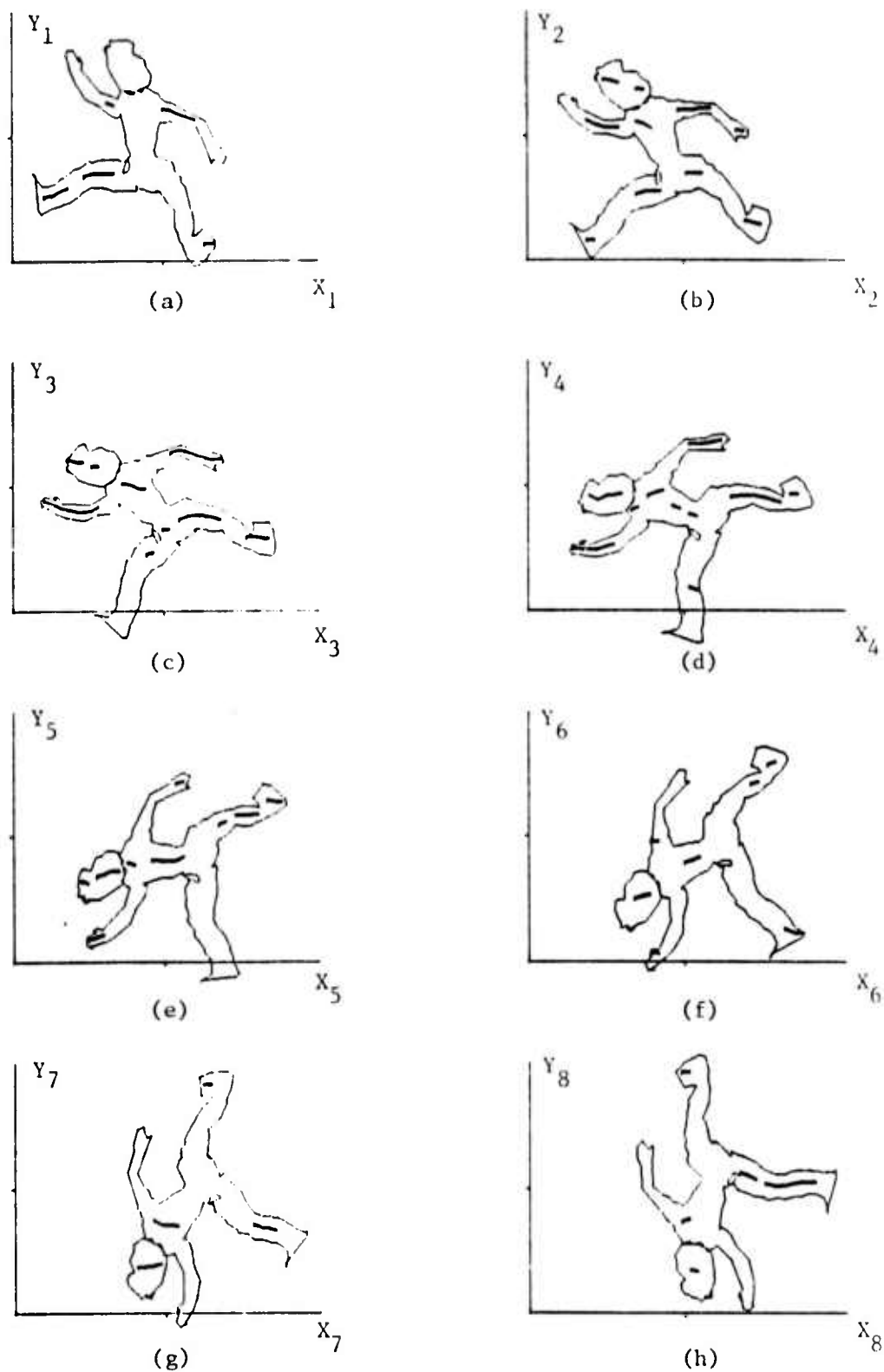


Fig. 4.4. Local Cones Generated by Projection in Eight Directions

Axis Refinement:

The axis refining process is iterative. We start from the list of axis points for the local cones provided by the projections. A straight line is fit to these points to approximate the axis. We construct the corresponding cross-sections, normal to the axis at the axis points, by finding intersections with the boundary. Only the end-points of these cross-sections are computed. The 3-d positions of these end-points are used. The distances from the axis of the mid-points of these new cross-sections are computed. If the mid-points lie on the axis, then the axis and the cross-sections satisfy the requirements of describing a cone. For real data, this requirement can only be expected to be satisfied approximately. We accept an axis, cross-section description, if the average distance of the mid-points from the axis is less than a threshold. (This threshold is set to the 3-d distance corresponding to 2 raster units, and is related to the expected random variations in determining the mid-points.) If the threshold is exceeded, we fit a straight line through the mid-points of the new cross-sections to define a new axis and iterate. The number of allowed iterations is set at 5 (we accept the resulting axis, cross-sections after 5 iterations). This process diverges when the axis direction changes to the extent that new cross-sections can not be generated by computing intersections with the parts of the boundary that the process began with.

Convergence of this iterative process is easy to see for a circular cylinder or cone for a wide range of starting directions. For a straight circular cylinder, consider starting with any set of parallel cross-sections, the line joining their mid-points immediately converges to the desired axis. Similar convergence follows for a regular cone, but for more general cones the precise convergence criteria have not been worked out. Empirically, the described process has been found to converge for elongated parts. When convergence fails, it is concluded that the part has no good description as a cone with the axis in the prescribed direction. This part may be later described as a cone with some other axis. Some areas may have no good descriptions in terms of cones and no cone descriptions might result for them. Description of such parts requires other primitives and is not considered in this work.

Cone Extensions:

Once an axis, cross-section description of a part is found, we try to extend the cone continuously over a larger part of the body. We extrapolate the axis at either end by a small distance (the choice of step size is discussed later). A cross-section normal to the local axis is constructed at this point and its intersections with the boundary are computed. If no intersections can be found extension terminates. (This indicates an end of the object or a sharp turning of the boundary).

Tests are made to determine whether this cross-section is acceptable as follows. The distance of its mid-point from the extrapolated axis is computed. If this distance is larger than a threshold (3-d distance corresponding to 4 picture units) then we make a modified guess at the extrapolated axis, by including this new mid-point and recompute a normal cross-section. (We have found it satisfactory to just accept the new recomputed cross-section and not iterate on this phase.) Fig. 4.5 shows an extended cross-section on a curving cylinder that is not acceptable, but provides a new direction for the axis and a new acceptable cross-section. This procedure allows us to trace the axis for a smoothly curving object.

The new cross-section is then tested for continuity of width with the previous cross-sections (the continuity evaluation is specified later). If the new cross-section is acceptable, further

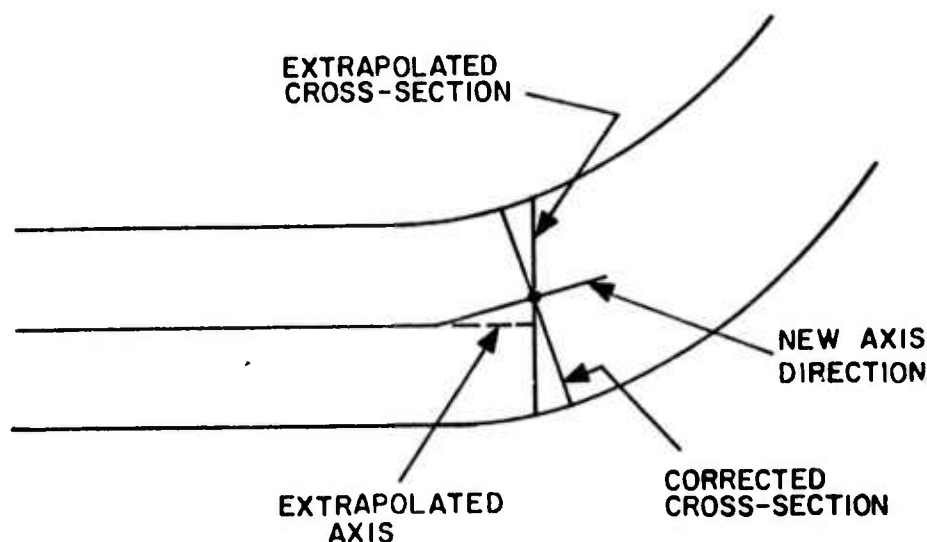


Fig. 4.5 Extension of a Cone

extension is attempted by iteration of the above described process. If a discontinuity is detected than the extension terminates. (Actually, before terminating, extension at half the initial step size is attempted.)

As example, when the local cones of Fig. 4.4 are extended this way, cones of Fig. 4.6 result (the axes of the cones are shown in this figure). Note the multiplicity of cones particularly for the head. For the other parts, various local cones have converged to nearly identical cones and are barely distinguishable in the figure. Each cone offers a potential segmented sub-part (choice of segmentations is discussed in sec. 4.5).

After termination of a piece, a check is made to see if the end of the object was also reached. We check whether the part of the boundary beyond the last cross-section is largely contained in a small extension of the cone. Part of the boundary may be beyond the last cross-section in the 2-d image but not in three-dimensional space. One instance of this is when a cone is terminated by a plane face. Example, see Figs. 4.7 (shows generated cones) and 4.8 (shows laser scans), the plane face on the hammer head is detected as a termination for the cone describing the head (piece P1).

This extension method is ad hoc, and the choice of parameters used determines how far a cone will be extended. Precise properties of the extended cone are difficult to determine and the success of the method has only been tested empirically. The following discusses some effects of the parameter choices.

The step size used in the extension process is important. We choose the step size to be proportional to the radius of the cylinder at this point ($0.05 \times$ the current radius, bounded by an absolute minimum and maximum step size). We expect to find elongated pieces and hence wider

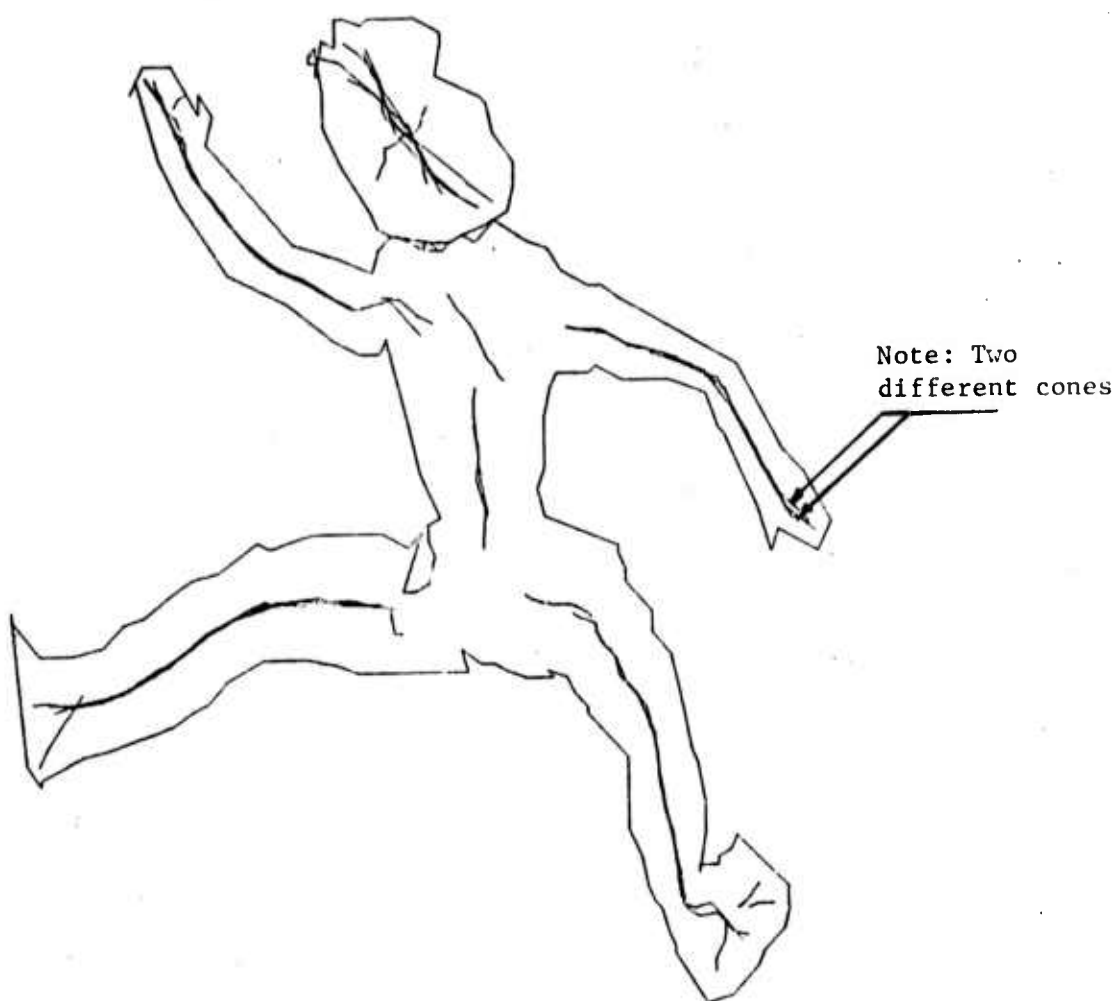


Fig. 4.6 Axes of Extended Cones for the Doll of Fig. 4.2

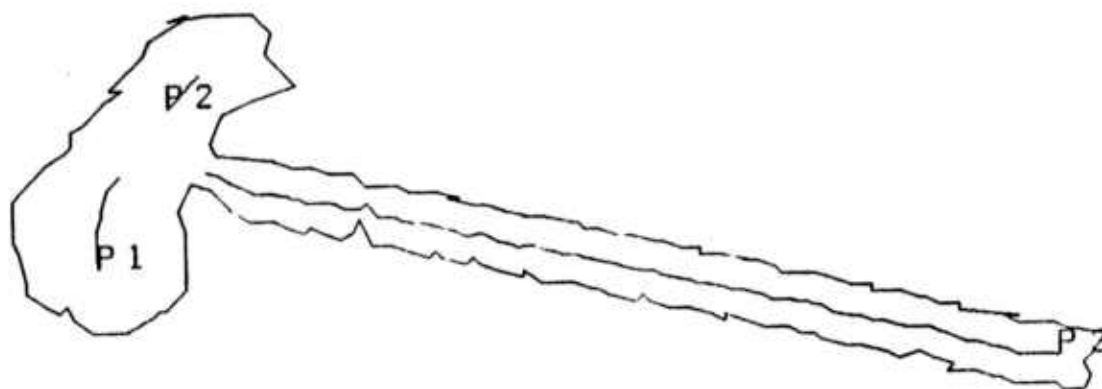


Fig. 4.7 Cones for a Hammer

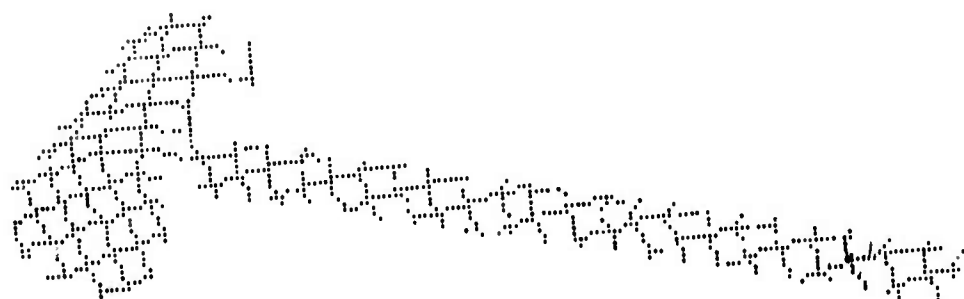


Fig. 4.8 Laser Scans for a Hammer

pieces are also expected to be longer. If extension fails we reduce the step size by a half and try again. If this fails too, we stop. Obviously, many smaller step sizes could be tried, at the cost of additional computation. The local axis direction is determined by fitting a straight line to a small number (5) of nearby axis points. These methods allow us to trace slowly curving parts of a piece, but may fail when the curvature is large. Higher level routines evaluate such segmentations and two disconnected parts may be connected based on context.

Evaluation of the acceptability of a new cross-section is designed to detect drastic discontinuities and context must be used for finer distinctions. A parabola is fit to a small number of previous cross-section widths and the width of the new cross-section is predicted. The actual width is allowed to differ from this predicted width by a fixed proportion (0.25). Boundaries constructed by our programs are frequently slightly jagged, because of misalignment of two laser scans and errors in computing segment end-points (see Appendix 1). This forces us to relax the continuity conditions for a cone, to avoid termination because of these small boundary fluctuations.

No explicit checks are made for the slope of axis to be continuous. If the cone curves too sharply, we find no boundary intersections for the extended cross-section and the cone terminates. Thus cones with an elbow, e.g. a human arm (see sec. 5.2 for elbow joints), will be segmented at the elbow depending on the curvature of the axis there. The next level programs are able to discover a segmentation at an elbow, and generate an alternative description merging the two (sec 4.5). The converse process of segmenting a cone at the elbow has not been used in our system; its implementation is direct.

The thresholds for cross-section continuity and step size were picked intuitively and adjusted empirically. A more analytical approach to such choices is not clear. Perhaps, single thresholds are not sufficient and alternate descriptions with different thresholds would be helpful. However, we believe that a general program should be insensitive to the choice of such thresholds. At least a partial solution is in the use of wider context for making segmentation decisions, such as later merging of two pieces separated at this level.

Choice of such segmentation criteria is a general problem that occurs in many domains e.g. linear approximations to a curve. We can do a better segmentation if we can look at the whole data globally, rather than just use local continuity criteria. We will then be able to make some use of context in deciding on the segmentation points. In the present case, this may be accomplished by using very loose constraints in cylinder tracing and then further segmenting the resultant piece. We may use the techniques commonly used for fitting straight line segments ([Duda, chapter 9]), to the axis and to the cross-section function. Usually, these methods attempt to keep the maximum error within a specified bound. Further segmentation decisions are meaningful only in the context in which they are to be used and hence must be made by the routines that use this segmentation data.

Extensions are found for all local cones suggested by initial segmentation. Thus many parts of the body will be included in more than one description. This allows us to compare alternatives and choose on the basis of wider context (see section 4.5).

These cone description routines only need to compute the end points of the cross-sections. The computation of the interior points of the cross-sections from the surface range data is straight forward. We have not used these interior points because of the difficulties in using

detailed description of the cross-sections in our system. Only part of these cross-sections is seen. The light source and the camera axis form an angle of 30-60 degrees, limiting the visible part of a circular cross-section to 120 degrees. Also, the cross-sections are small for objects of a size conveniently used with our scanning apparatus. The visible part of these cross-sections is nearly flat and the ranging errors become significant in the description of their shape. We use perceived width as the basic descriptor.

Some summary descriptors are used to describe the gross shape and the size of a piece. We have used the length of its axis, the average cross-section width, the ratio of the length to the width and an average cone angle. More details on symbolic descriptions of the pieces are presented in section 5.1.

Performance:

We present results on some scenes that were also used by Agin in previous work ([Agin 72]). Figs. 4.9 and 4.10 show the results of our programs on a glove and a horse. Agin's description programs merged parts of the index finger and the little finger in the glove, and the body of the horse extended to include the tail (we have not reproduced the pictorial results of Agin's work). Our description methods join the various cones to form complete objects as described in chapter 5, whereas Agin's descriptions only described isolated parts. Our programs are also substantially faster, as we need to work only with the boundary of an object. Also, our methods do not assume any particular cross-section shapes, whereas Agin's methods were restricted to circular shapes.

Our programs give satisfactory performance on scenes of complexity illustrated here (more examples will be presented in chapter 7). If a sub-part is elongated, it is well described by a cone and our programs are usually successful in finding such descriptions. Some extensions, such as when other sub-parts interfere, cannot be made properly without the use of this contextual information (sec. 4.5). This advice is best supplied from the higher level routines. A weakness of the programs is the failure to verify that the computed cone description in fact describes a cone. We may describe a plane surface as a surface of a cone whereas it may be better viewed as a termination of some other cone or just a flat surface. Design of special routines to detect such cases needs to be investigated.

4.4 OTHER CONE DESCRIPTION METHODS

Iteration is used for accurate location of the axes of cylinders in the methods of the previous section. An alternative to iteration is to find a best cone that fits the given boundary segments. The cone axis must be constrained to a certain form such as straight or parabolic, and the cross-section function may be limited to be linear locally. A best fit in the least square sense, with these constraints, will give us the axis and cross-section function directly.

Differential Geometry ([O'Niell]), is concerned with descriptions of surfaces located in three dimensional space and may be relevant to the generation of the desired cone descriptions. It characterizes surfaces by a small number of variables, mainly using local curvature. The author briefly investigated the use of principal curves, which are the directions of minimum and maximum curvature. The temptation of using the principal curves lies in the fact that they are

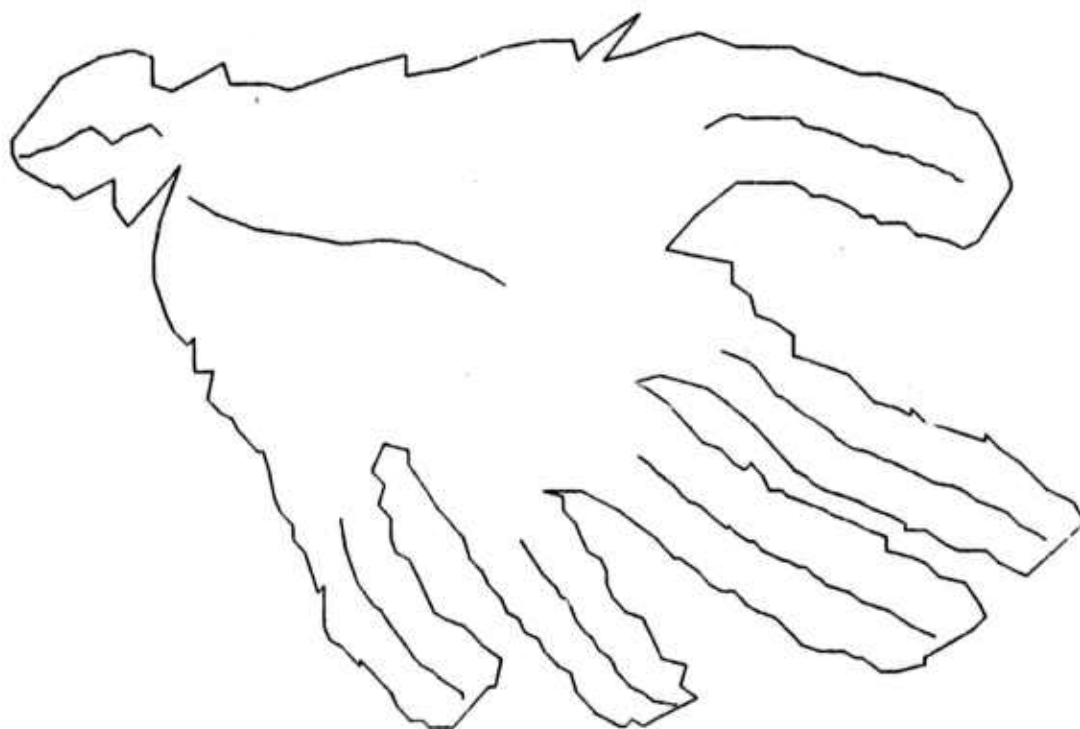


Fig. 4.9 Cone Descriptions for a Glove

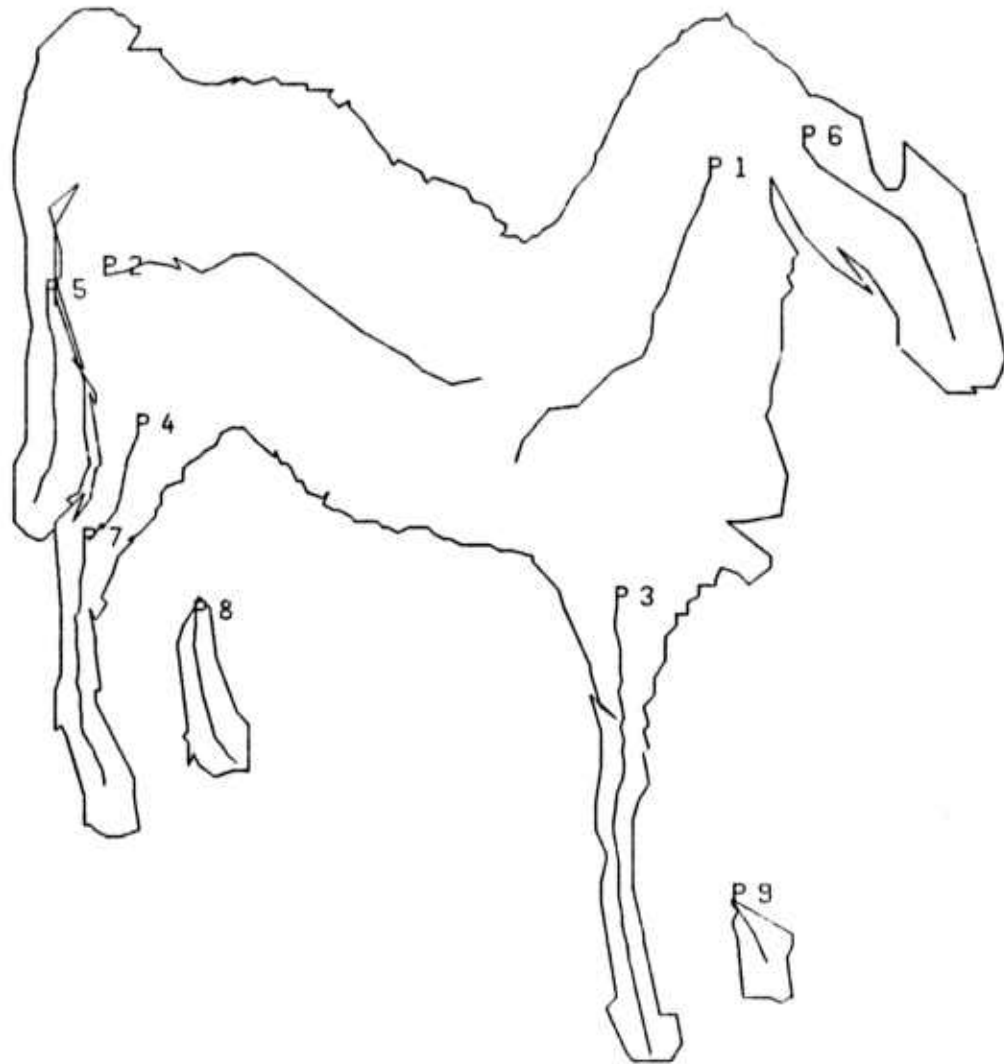


Fig. 4.10 Cone Descriptions for a Horse

intrinsic to the surface, and are not sensitive to obscuration or viewing angles (the visible part of them of course is). For the case of circular cylinders and cones, maximum curvature curves coincide with our choice of cross-sections. However, for more general cones, no such simple relations exist. E.g. for an elliptical cone, the maximum curvature curve is not even planar. (This curve can be easily constructed by observing that it must be normal to the generating line in the tangent plane, as two principal curves are always orthogonal. The principal curve goes towards the apex starting from the major axis and returns after reaching the minor axis).

Computation of curvature requires the use of second derivatives of the local surface, and can be strongly affected by ranging errors. We experimented with programs to search for directions of minimum curvature by defining "average" curvature over a short distance. The average curvature along a given direction over a chosen length was computed by measuring "bending" defined as the maximum excursion of surface points in between, from the line joining the given end-points. Moderate success was obtained for those parts of an object that were elongated. For such parts the curvature changes slowly, and the measurement of average curvature is better justified.

The author was unable to find any simple relations between the chosen generalized cone representations and the entities used in Differential Geometry. For this reason and the above stated difficulties of computing derivatives, the subject was not pursued further. However, the principle curvature directions may still be useful in choosing among alternate descriptions, e.g. in choosing a preferred axis direction for a thin disk (the elongation direction is *not* the preferred direction here).

4.5 CHOICE OF SEGMENTATIONS

The cone description routines generate a number of possible sub-parts. Many of these cones share common parts of the object and hence are not all compatible with one another (e.g. see the various cones for the legs and the arms of the doll in Fig. 4.6). However, some segmentations are more appealing, intuitively, than others. We aim to choose a small number of segmentations into sub-parts, the sub-parts in one segmentation being compatible. We prefer cylindrical and elongated descriptions. Pieces with length to width ratio lower than a threshold are discarded, provided they overlap with some other pieces.

The simplest form of overlap occurs when a part is described more than once because the cone extension programs starting from different local cones, converged on nearly the same cones. E.g. see the two cones describing the left arm of the doll in Fig. 4.6 (the axes of the two cones overlap a lot and are barely distinguishable in the figure). The various descriptions here are equivalent and we may choose any one. We prefer the longest cone. Some cone descriptions are terminated prematurely, due to a local discontinuity of the boundary. Another cone describing the same area may not be broken. If a cone is completely or largely included in another cone, we retain the containing cone only.

Another class of conflicting descriptions is caused by the effects of corners. Consider the two dimensional example of a rectangle, Fig. 4.11, showing axes of various cones by dashed lines. It may be described by cylinders with axes along the sides, or by axes bisecting the corners. Among the cylinders with axes along the sides, we choose the more elongated one, i.e. one having

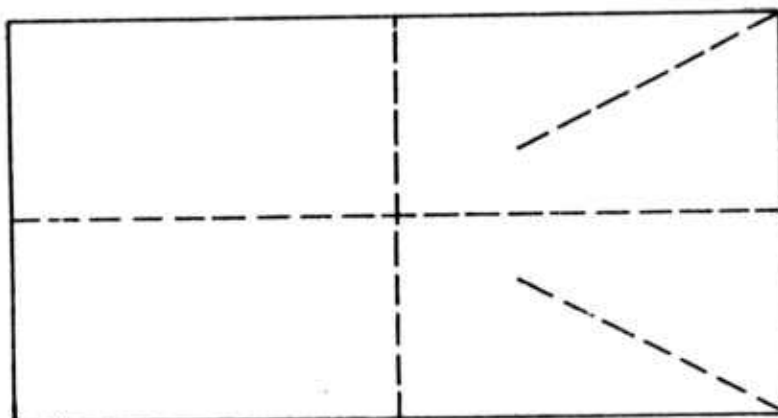


Fig. 4.11 Different Cones for a Rectangle

the larger length to width ratio. The "corner" cone is included in the longitudinal cylinder, or small extensions of this cylinder. The latter case occurs when another cylinder is present near the corner and prevents the longitudinal cylinder from extending to the end. We choose to ignore the description which is contained in the other. More details of these computations are provided near the end of this section.

Sometimes, no clear choice can be made between two alternate piece descriptions. E.g. the head of the doll in Fig. 4.6 is nearly spherical and many axes directions are equally good. In such cases we may retain the various alternatives and make multiple descriptions (this example should really be described by a sphere primitive). Current programs pick one of the alternatives only, but the data structures allows easy addition of alternates. (Note the following describes an instance where we do use alternate descriptions.)

Application of the above selection criteria results in the selected segmentation of Fig. 4.12 for the doll of Fig. 4.6 (the algorithm is stated precisely in Appendix 2). Note the parts in Fig. 4.12 are numbered in an arbitrary order. Both arms and legs are described by more than one cone each initially. Note that the small cones describing the feet of the doll were computed to be contained in the leg and do not appear in the final choice. Cone P5 describing the top of one leg was not judged to be included in the extension of P6 (but an alternative description merging the two is also generated as described in the following). The shoulder piece in Fig. 4.6, is contained in the extensions of the body piece and hence does not appear in Fig. 4.12. Such computation is not very robust. Shortly, we describe a technique of redescription which would be more reliable.

Use of local discontinuities for making termination decisions in the cone description process results, sometimes, in premature termination. A part is thus broken into two parts separated by a small discontinuity between the two parts. If such two parts extend into each other

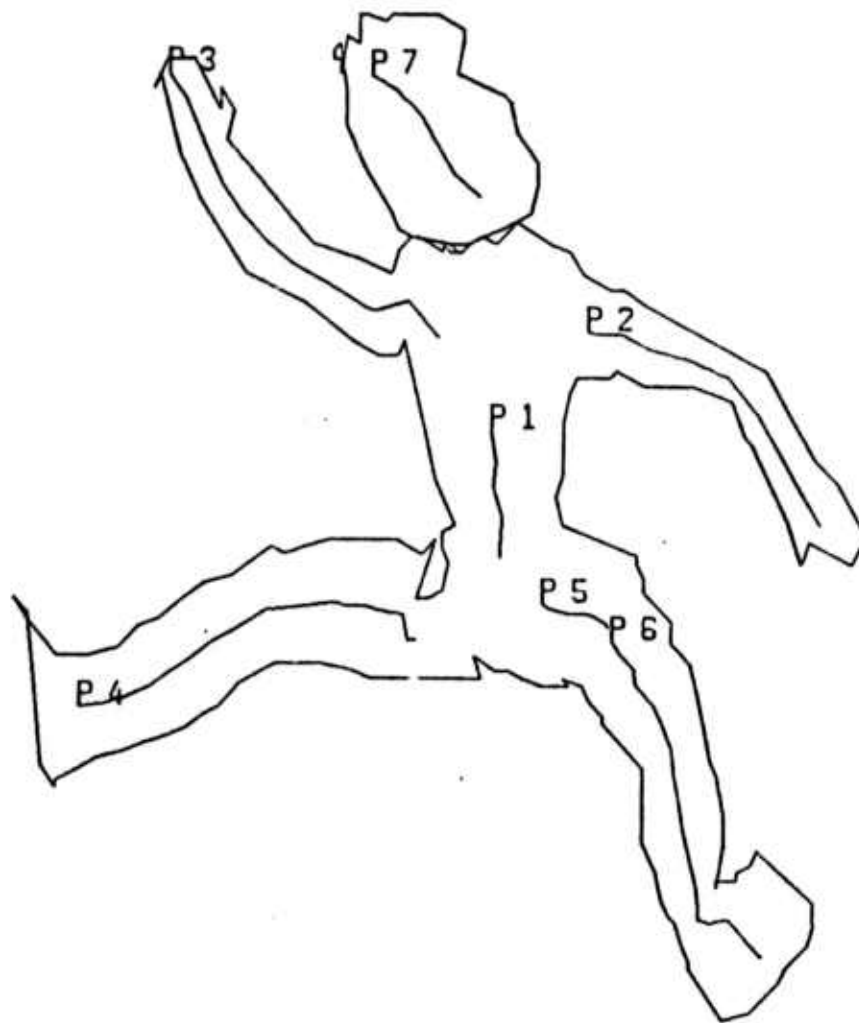


Fig. 4.12 Selected Cones from fig. 4.6 for a Doll

continuously, we generate an alternative description that creates a new part combining the two parts. For example in Fig. 4.12, P5 and P6 are merged into a single part. Such decisions to consider this entire volume as one piece can not be made by our primitive description routines, as they look for local discontinuities only, and have no notion of an isolated, small discontinuity. (This merging actually takes place after "joints" of selected pieces have been computed. Joints are discussed in sec. 5.2 and the details of merging in section A2.2B of appendix 2.)

Extension of some pieces is terminated by the interference of other pieces attached to this piece. The extended cross-sections begin to include parts of the other pieces. For example in the description of the doll in Fig. 4.6, the piece describing the body of the doll does not extend into the shoulder area, the extension being inhibited by the presence of the arms. Descriptions of such pieces can be improved by redescribing the cone primitives using a modified boundary, generated by "cutting off" interfering pieces. This redescription technique has not been implemented; use of the boundary by the cone description routines permits a direct implementation. Some redescriptions of primitives may also benefit from being guided by more specific information provided to the cone description procedures, such as a prescribed axis direction or the cross-section widths.

Another example of alternative segmentations and chosen segmentations is shown in Figs. 4.13 and 4.14. More examples of final segmentations are presented in chapter 7.

In the rest of this section are presented some details of the programs used for selection of segmentations.

The extent of a piece is defined by the boundary segments of its two sides. We use the boundary segments to determine the amounts of overlaps of two pieces. Boundary overlaps are easier to compute, but not as closely related to the desired geometrical computations as area overlaps. Eg. in Fig. 4.13, the doll body is described by two cones with nearly orthogonal axis; these cones share substantially the same area but no common boundary! (Correct choices were made in this case, only because one of the pieces was discarded due to its very low length to width ratio.) Normally, the area computations require significantly larger amount of computation than the boundary computations. However, when evaluating the area overlap of two cones here, the approximations of the cones by their axes points and cross-sections are already available, i.e. the areas to be compared have been segmented in sequential trapezoidal strips. This reduces the complexity of the required computation. We have used boundary overlaps, but judge the implementation of area overlaps necessary for increased performance in further work. For this implementation, a cone is considered to be completely included in another if a large part (> 0.75) of boundary segment of its two sides is included in the other.

If complete inclusion does not occur, the maximum distance of one conflicting piece from the other is computed. If this distance is a small fraction of the length of the including piece, then inclusion is assumed. If only one side of a piece is included in the other, but the piece is not elongated or is very conical it is eliminated.

These overlap resolving methods are simple but work well for our examples. We expect significantly better performance if the techniques of redescription by removing some parts and the computation of cone overlap by using areas instead of boundary were to be used. Addition of the redescription techniques would not require any modifications of the cone description routines, they need to be simply supplied with modified boundaries. Interfering parts can be easily determined

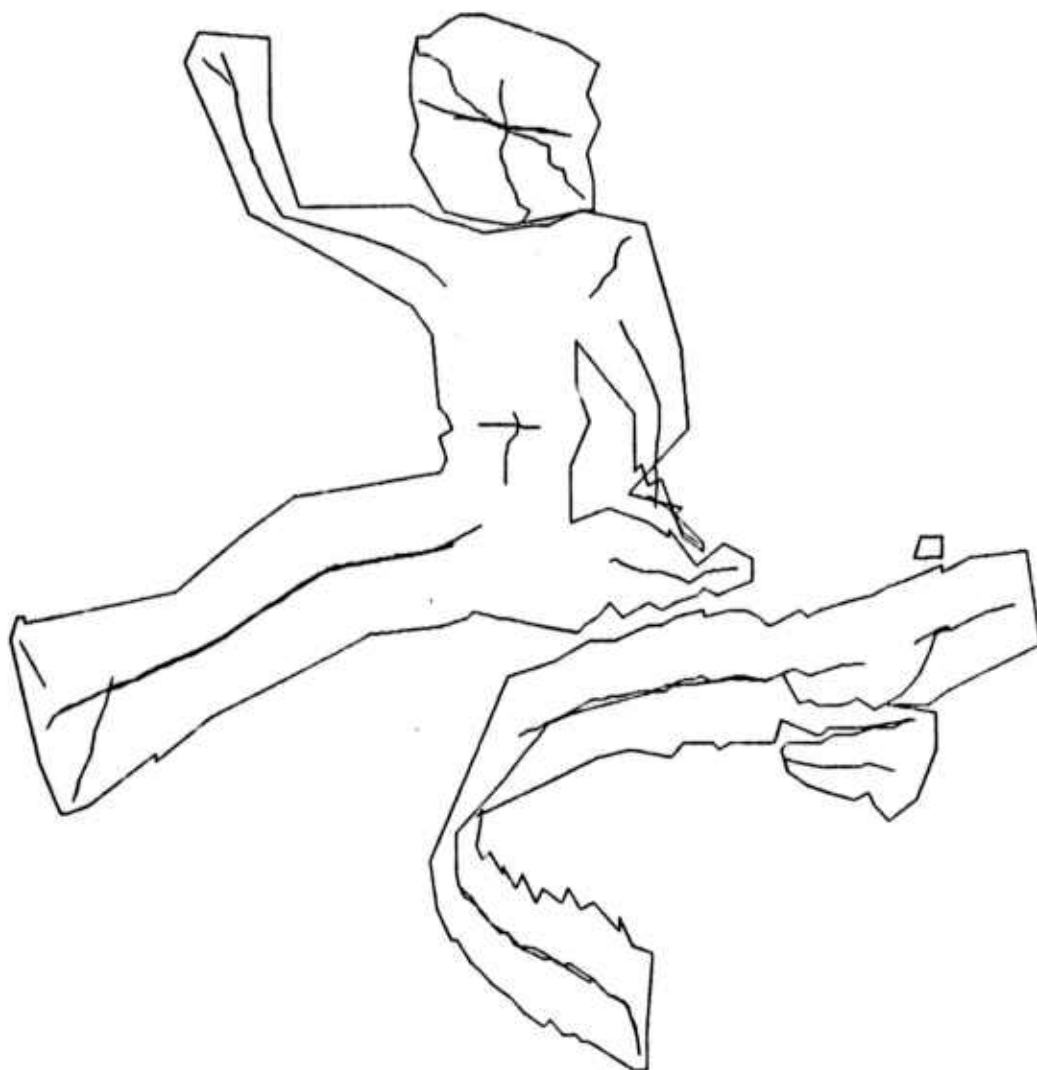


Fig. 4.13 All Extended Cones for Scene of Fig. 3.5

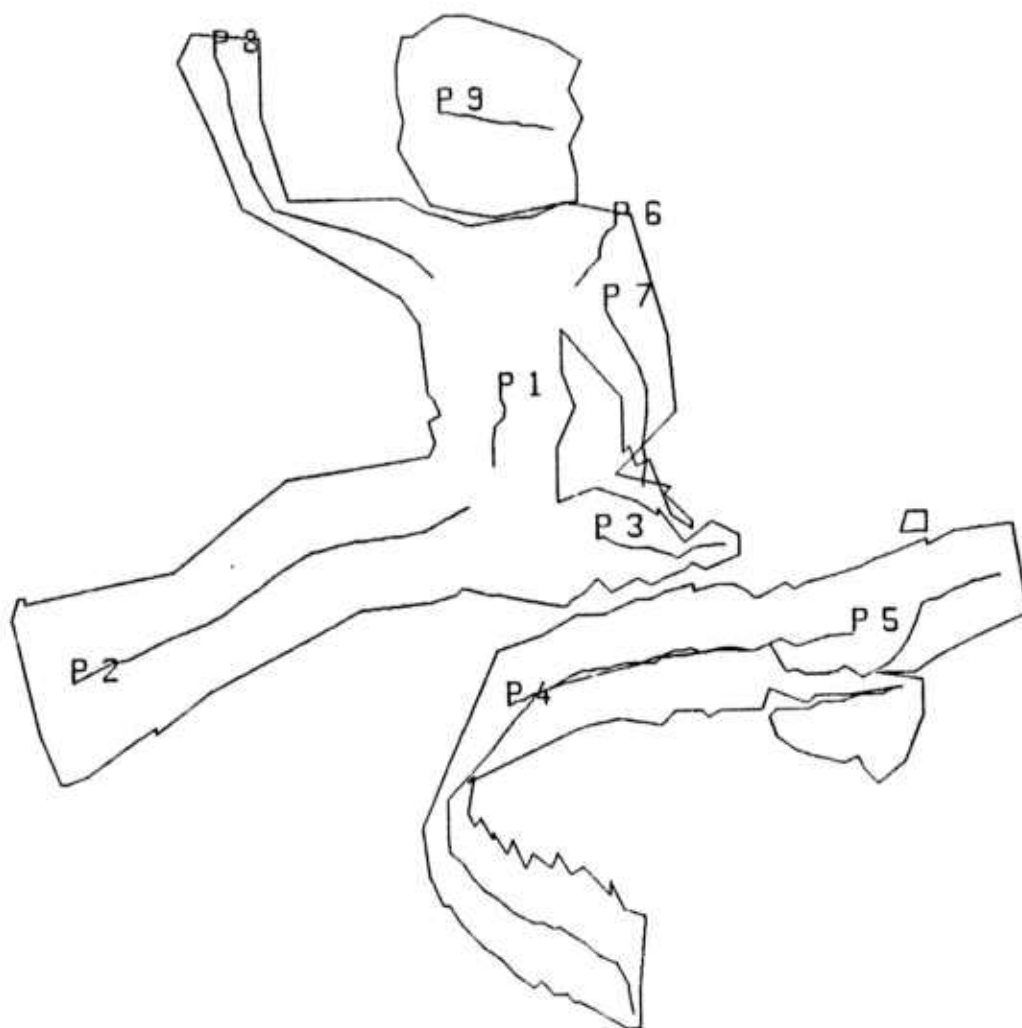


Fig. 4.14 Selected Cones from Fig. 4.13

when the joints for this piece are computed (sec. 5.2). However, we must make judgements about which pieces are already well described and can be cut-off, and which others need redescription.

4.6 OTHER DESCRIPTIONS OF PRIMITIVES

We have discussed the description of objects by generalized cone primitives. Some objects or some parts of these objects are not well described this way and other primitive descriptions must be used. These additional primitives may be used in conjunction with cone descriptions or may be completely independent. We will discuss only a few additions that may be found useful in further work.

1. Planes: Planar surfaces are frequently present in machined objects and are not necessarily parts of cylinders. Sometimes, they occur as terminations of cylinders and our methods are able to notice this, if the terminations are nearly orthogonal to the axis. More generally, we should identify the planar parts of the surfaces and evaluate whether they form terminations and their suitability for being described as parts of cylinders.

2. Spheres and Bowls: Spherical objects may be described by an axis, cross-section description, but no preferred axis directions exist and it is simpler to describe them as spheres.

3. Holes: Holes may be described by describing the volumes enclosing them or sometimes more conveniently as negative volumes. Description of negative volumes is the same as that of positive volumes and the same description methods apply. In particular, holes may be described as negative cylinders.

CHAPTER 5

SYMBOLIC DESCRIPTIONS

From the chosen segmentations of an object, we aim to generate structured, symbolic descriptions attempting to capture the "important" shape characteristics of the object. The descriptions contain enough information for recognition of the objects and for indexing into a visual memory for similar objects. Other uses for these descriptions may be in computations for manipulation, stability or for acquisition of models. We consider the ability to generate useful symbolic descriptions as the central issue of this thesis. Utility and performance of programs for applications, such as recognition, depend directly on the ability to generate (and manipulate) these descriptions.

The segmented parts connect at *joints*. The joints and the parts determine the structure of the object. Description of this connectivity structure is a major component of the object description. Object descriptions contain descriptions of the parts, the joints, and their relationships, using global as well as local properties. The local descriptors provide details of the individual parts, and their shape. Joint descriptors depend on the angular, positional and the size relations of its constituent parts. Global descriptors depend on relations among larger number of parts, e.g. the detection of bilateral symmetry in an object structure.

A major goal of these descriptions is to aid in recognizing an object as belonging to a class of models, and to be able to make detailed comparisons within a class. The structure is the most important descriptor used. The details of the parts are used to make finer distinctions. The descriptions are hierarchical; varying amounts of detail can be added to the basic descriptions. Our descriptions of the joints and the pieces are limited because of the problems of using descriptors that are not invariant with viewing angles and limb articulations, and because of the need for better low level (cone description) routines to allow better symbolic descriptions.

In our system, only cones have been used for describing parts. These cones are allowed to attach to the other parts of the object, to form joints, at each end only (no joints along the sides of two cones are allowed). Such joints can be defined by an area not included in any piece description. A number of pieces are connected to a joint. Connectivity of various parts is easily inferred from the boundary. Connectivity relations are a very central part of the descriptions of an object. Some parts are partly shadowed and their connections to other parts are not directly known. They appear isolated and we must hypothesize their connections. (E.g. two legs of the horse in Fig. 4.10.) We have implemented simple hypotheses mechanisms, other mechanisms are suggested (see section 5.4). These hypotheses are further examined in the process of matching with models.

Different pieces and joints are represented as symbolic entities. Connectivity relations of the object may be viewed as graph relations with joints as nodes of the graph and pieces as the arcs between them or vice versa. E.g. the graph structure for the doll of Fig. 4.12 is shown in Fig. 5.1. (This graph shows pieces P5 and P6 as merged into one. The "B" and "H" pieces are *distinguished* as explained later). The graph shows the two arms and the head of the doll joined to the body at one end and the two legs joined to the other end. (Note, the information about connection of the head to the body is missing in Fig. 4.12; the graph shown here is idealized.) Various descriptors are attached to both the joints and the pieces. Relations other than connectivity also exist between various pieces and joints. Data structures for symbolic descriptions are presented in Appendix 3.

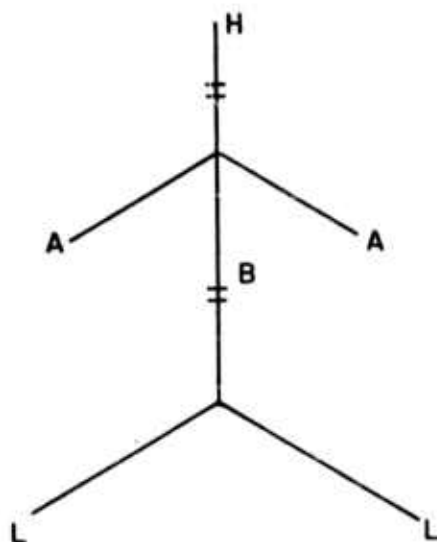


Fig. 5.1 Connection Graph of a Doll

Descriptions are generated for the pieces and the joints of an object. Details of piece descriptions are limited by the rather sparse data that is available for cross-sections. The axes of pieces are not so accurate near the ends of the pieces, because of the effects of other connecting pieces; this affects the accuracy of the piece descriptors and also the joint descriptors which rely on the angle between the pieces.

Major descriptors for a piece are based on its relative size. We use the length of its axis, the widths of the cross-sections, and the ratio of the axis length to average cross-section width (elongation). The cross-section function is approximated linearly. This is equivalent to specifying an "average" angle for the apex of the cone. Some qualitative descriptors such as straight axis, conical or cylindrical, regular or irregular are computable directly from the piece axis and cross-section data, but have not been used.

Joints are described by the relationships of the parts attached to them. Angular, positional and relative size relations of the parts are used. The number of pieces at a joint and their relative sizes are noted. The *type* of a joint is described, for example a T-joint or a neck joint. However, these descriptions are not invariant to limb articulations, making their utility for recognition programs very limited. The different joint types aim to reflect different possible physical constructions for the joints, but the inference of the constructions from the descriptions is difficult.

The global descriptions aim to describe the important characteristics of the whole object or of some large portions of the object. It is common for a large number of narrow parts to attach to a wider part, in natural as well as man-made objects. E.g. the arms and the legs of a doll attach to the body. As a consequence it is useful to consider a size hierarchy based on the widths of

the cross-sections of a part. The largest part is at the lowest level and attached parts at successive higher levels. The matching process need only match pieces at the same level. (In an occluded scene, only the relative levels may be known. Then matching between different levels has to be tried, preserving the relative order.)

Our implementation of size hierarchy is in computing the large pieces of the object and considering these to be *distinguished*. Other distinguishing characteristics for a piece can be defined: we use the property of a piece being very long, in comparison to other pieces of the same object, e.g. the handle of a hammer. These distinguished pieces help in making immediate correspondences between two descriptions during matching for recognition; the matching process starts by matching pieces with similar distinguishing characteristics.

Planes of bilateral symmetry for an object are searched for. Tests for similarity of two pieces are based on their gross properties. Knowledge of distinguished pieces helps in limiting the search for symmetry planes, as these pieces have few or no symmetrical pairs. Symmetry computations are complicated by occlusion. Some parts may be hidden completely or partially.

Appendix 3 includes the details of the data structures used for symbolic descriptions.

5.1 PIECE DESCRIPTIONS

Summary descriptions for a piece are used to describe the size and the gross shape of a piece. Such descriptors, both qualitative and quantitative are useful for quick, crude matching of two pieces. These descriptors are computed from the more detailed axis, cross-section descriptions. The axis has been represented by a list of points that define a space curve and the cross-sections by their end-points. This descriptive information can be used for point to point matching of two pieces.

The important size summary descriptors used are the axis length, the average cross-section width, and the ratio of the length to the width. Elongated pieces (length to width ratio > 3.0 say) are of particular interest. They are "well defined" and unlikely to appear spuriously in descriptions. A linear approximation to the cross-section function is made. This corresponds to fitting a linear cone to the piece and the cone angle is used as a descriptor. Matching procedures use the cone angle to differentiate between cylindrical and conical pieces. We also retain information about the ends of a piece, consisting of the location and the direction of the local axis, and the local widths of the cross-sections.

In the following we suggest some techniques that would be useful for further descriptions of the cone axes and cross-sections. These have not been implemented in our system and the reasons for not using them are given.

The axis of a part is a curve in three-dimensional space and normal curve description techniques are applicable to its description. The axis may be approximated by a set of curves, such as straight lines or splines. Choice of the segmentation points, or the positioning of the knots in the spline fit case, is crucial to good description. Segmentation points should be at discontinuities; points of inflection and high curvature change are obvious choices.

The cross-sections are a planar area, they may be described by segmentation of this area into two dimensional "cone" primitives, in a manner analogous to the segmentation of three dimensional objects. In fact, the same programs can be extended to handle the two dimensional case. Such segmentation would permit us to handle complex shapes for cross-sections with discontinuities and corners, as in a fluted cross-section, for example. The cross-section function, i.e. the function describing the changes in the cross-sections along the axis can also be described by segmentation, say by piecewise linear functions. Currently, we approximate the cross-section function by a single linear function.

Such techniques for describing the axes and the cross-sections are useful. However we have not concentrated on these, believing that they represent independent numerical analysis problems, not central to the problems of object description. Also, the data about the cross-sections in the present implementation is limited, and the segmentation procedure is likely to be unreliable. Some gross shape properties about the axes, and cross-sections, such as straight axis, conical or cylindrical part, regular or irregular, convex, flat or concave cross-sections can be obtained directly, but many of these descriptors require the use of a threshold judgement, e.g. between a straight and a curved axis. We have not used such descriptors either.

5.2 LINKING OF PIECES AND JOINT DESCRIPTIONS

In our representation, we are interested in describing the joints between different parts. The description of parts has been restricted to generalized cylinders; we further restrict a cone to join the other parts at one of its two ends only. This is rather restrictive, e.g. a hammer cannot be described as a handle connecting to the middle of the hammer head, but the head needs to be described as two pieces which connect at the handle. However, this restriction has not been very important for the objects considered here and leads to a very simple algorithm for connecting pieces. Other ways of joining parts can be easily added.

For this situation, a joint is adequately defined by an area which is not included in any piece descriptions, and by the pieces which adjoin this area. The construction of a joint area is very simple using the boundary: we start at one end of some piece and move along the boundary until we come to another piece; then skip across the new piece and continue along the boundary in the same way until we have returned to the starting point. Our path defines the joint, it consists of ends of pieces and boundary segments between them. These boundary segments are null if the adjoining pieces overlap. In an extreme case, where all connecting pieces intersect, the joint area may be zero. We find new joints until ends of all pieces that do not terminate at the end of an object have been included. Fig. 5.2, shows the joints obtained for a doll, from the pieces shown in fig. 4.12, in this way (the joint areas are shown shadowed). Note that the joints J1 and J2 are between pieces that do not overlap; joint J3 is between two overlapping pieces and joint J4 has only one attached piece (this piece failed to extend to the end of the leg).

Symbolic descriptions of a joint contain the order of pieces connected at a joint and a dominant piece, which is the widest piece of the joint. The order of pieces is not invariant to viewing angles since the parts are connected on a two dimensional surface, which does not have a useful invariant ordering. However, for many objects, particularly when the parts occur along a plane curve, the order is preserved for other viewing angles; our recognition programs assume this order preservation.

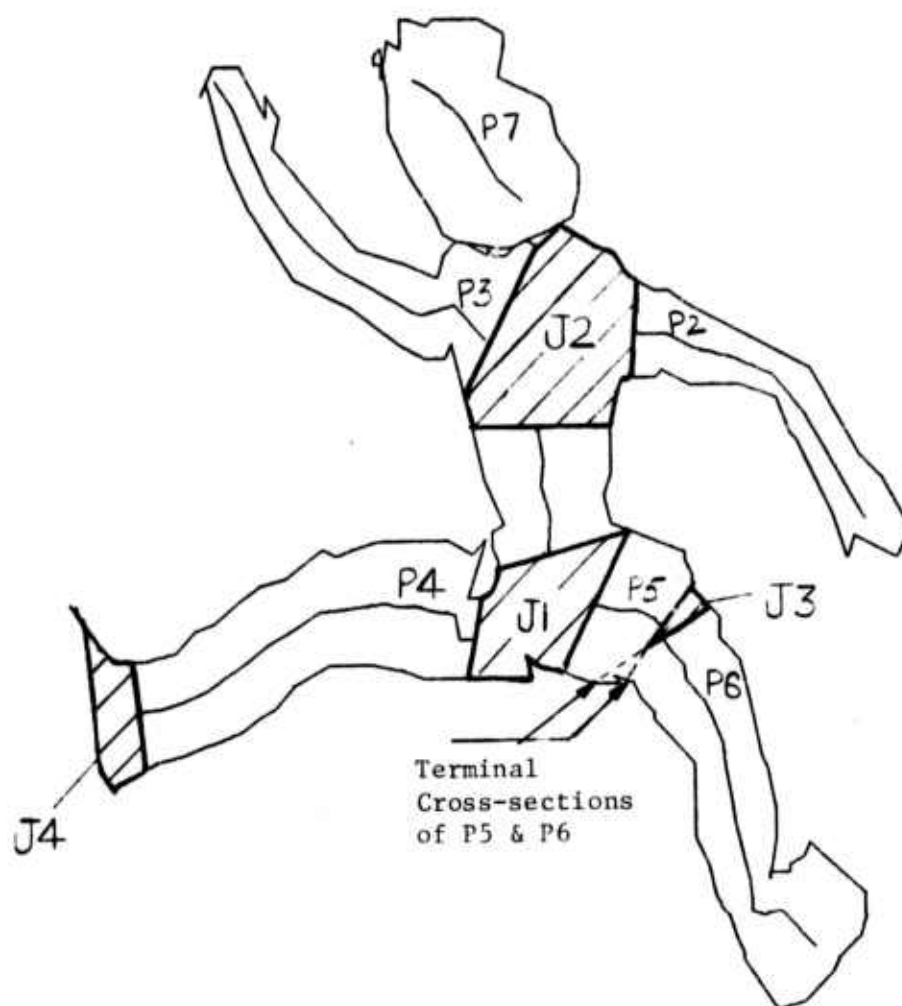


Fig. 5.2 Joints between Pieces of the Doll of fig. 4.12

In the following, we describe other joint descriptors that were implemented but have not been used in recognition for reasons stated later.

We describe the joints by "types", however these classes are not mutually exclusive, and new type additions can be made without affecting the old ones. We have tried to define joint descriptors that correspond to different ways the joints are formed in the physical construction of objects. However the inference of the construction from the descriptions is not well understood.

A catalog of various descriptors and their definitions follows:

T-joint: Two pieces are collinear and continuous and connected to a third, non-collinear piece. The two collinear pieces are allowed to appear merged in a single piece (Fig. 5.3a). An example is the joint of a hammer handle to its head.

Fork Joint: One piece is "opposite" all other pieces, i.e. a half-plane separates this one piece and the others (Fig. 5.3b). An example is a human hand.

Neck joint: Two pieces with different cross-sections, but with axis continuity (Fig. 5.3c). An example is a human neck.

Elbow joint: Two similar but non-collinear pieces (Fig. 5.4d). Examples are human elbow and knee.

Cross-section Conservation: between a large piece and attached smaller pieces (Fig. 5.3e). E.g. both human leg cross-sections are conserved at the torso.

Coplanarity: All constituent limb axes are coplanar.

Programs for computing the joint descriptors follow the above definitions directly. However, some approximations must be used, partly for lack of complete information in a scene and partly to be insensitive to the errors of low level descriptions. The axis directions are ill defined near the ends of a piece; hence angular judgments are inaccurate. We see only part of the cross-sections and use the perceived width for those descriptors that need cross-section information.

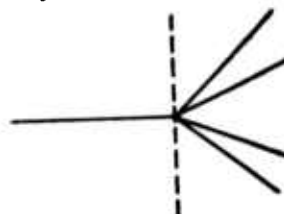
Collinearity of two pieces must be computed for T-joint descriptions. We use the continuity of axes directions and cross-section widths. Continuity of boundary near and between the ends of the two pieces would provide a better continuity check (the boundary on one side of these two pieces must be continuous for a T-joint). Loose constraints (upto 30 degrees) are used on axes directions for determination of collinearity.

Of the above joint types, the fork and the coplanar joints are dependent on the limb articulations. Also, our cone description routines do not always provide satisfactory axes directions near the joints. These descriptors would be of obvious value for non-articulated objects or in some cases where the articulation limits were known. Determination of cross-section conservation requires seeing all the pieces of the joint. These reasons have prevented the use of joint types for recognition.

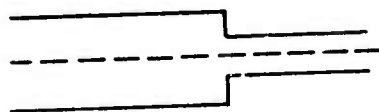
If the object is known to have a T-joint formed by connection of one piece in the



(a) T Joint



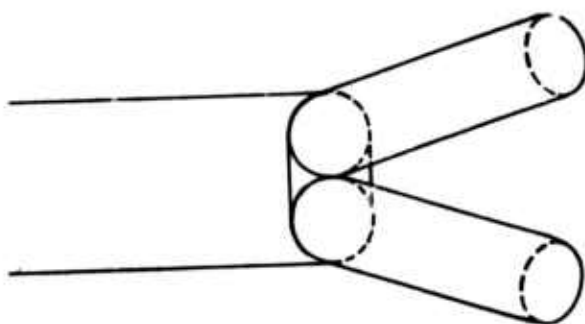
(b) Fork Joint



(c) Neck Joint



(d) Elbow Joint



(e) Cross-section Conserving Joint

Fig. 5.3 Different Joint Types

middle of another piece, then this descriptor is invariant and could be used for recognition. In the current implementation, our models do not have such information.

5.3 OBJECT DESCRIPTION

The descriptors used for individual pieces and joints have been discussed; we also make descriptions for a complete object. These descriptions aim to capture the important features of many pieces and joints of the object and their relationships, and be useful for recognition. The connectivity relations between pieces and joints are known; we have a description of the object that may be viewed as a graph structure. We now aim to characterize the important parts of this structure.

Simple descriptors are the number of joints and the number of pieces. The number of elongated pieces (length to width ratio greater than a threshold) is also used, since it is less sensitive to segmentation differences (but still subject to variations caused by occlusion).

The large pieces in an object description are *distinguished*, as there are only a few of these. (If limbs are formed by a large piece splitting off into many pieces, it results in a clear discrimination between larger and smaller pieces.) The pieces can also be distinguished by other characteristics, we have used the property of a piece being more elongated than the others, e.g. the handle of a hammer. Only similar distinguished pieces may be matched for recognition, allowing for efficient matching between two descriptions (see chapter 6).

We note whether a distinguished piece has pieces attached on both ends; their number, and their sizes relative to the distinguished piece. If pieces at one end of the piece are very different than at the other, such as being at least twice as wide or twice as long, this description is also associated with the distinguished piece. This helps in "orienting" the distinguished pieces in matching.

Bilateral symmetry is often found in natural and man made objects, and planes of bilateral symmetry are useful descriptors. For a bilaterally symmetric object, sub-parts must occur in symmetrical pairs, unless they lie along the plane of symmetry. If the axes of all parts of an object are coplanar, this common plane is also a symmetry plane (front/back); such symmetry planes are not of much interest and are not described. The distinguished pieces of an object do not normally have a paired piece. In this case the symmetry plane must pass through the axis of this piece, or be normal to this axis and divide the distinguished piece in symmetrical halves.

The search for symmetry planes is confined to those passing through the axes of the distinguished pieces. At each end of the distinguished piece we look for symmetric pairs of limbs. The symmetry plane must pass between a symmetric pair. Once such a pair is found, the symmetry plane is constrained. If more than one pair of limbs is symmetric to each other, a few alternative planes are possible. We only need to test the symmetry of other limbs relative to these planes. This aids in determining the connections of shadowed limbs (Cf. sec. 5.4).

To evaluate object symmetry, we need to evaluate limb similarities. We have only simple descriptors for individual pieces. The similarity test is based only on the lengths and the widths of the limbs. Matching of two pieces by comparing their cross-sections at each point along

their axes would give better discrimination. Articulation of limbs is allowed; this articulation need not be symmetric, thus the bilateral symmetry computations are limited to using the attachment points. Some parts are shadowed and their attachment is unknown, we include them in symmetry evaluations using the hypothesized attachment points (Cf. sec. 5.4). If symmetry is found, it provides further weight for connections of the shadowed parts. However, similarity of the limbs in shadow to the other limbs is difficult to determine.

Our evaluations for symmetry are admittedly crude, however it is felt that when symmetry is discovered, it is not likely to be accidental and can be a useful descriptor.

5.4 LINKING OF SHADOWED PARTS

We have a partial body separation into groups of connected pieces. Some pieces have no connections, because of occlusion and shadows. E.g. two of the legs of the horse in fig. 4.10. Connectivity of these pieces cannot be known directly, we can only hypothesize possible connections. The shadow regions are known from the knowledge of the position of the light source. Clues for connections are obtained from proximity and symmetry. Semantic knowledge such as stability and support relations would be helpful. Some objects will not be stable without the connection of isolated parts, e.g. a horse cannot stand on only the right front and the right rear leg. The stability problem is difficult since only the front surface of the object is seen. We have not used stability considerations.

If multiple objects are present in the scene, we must estimate what body an isolated piece is connected to. Even if only one object is known to be in the scene, we must estimate its attachment points.

The light beam and the camera axis are not collinear. A point in space is invisible if it is occluded from either the camera or the laser view. Some improvements could be obtained by using a camera image in addition to the laser scan data. The position of the light source is known and the shadow regions are computable. The parts that cast shadow on other parts are also known. The connection of the isolated part to any other part must be through a shadowed or an occluded region. However, this does not uniquely determine the connections. Use of monocular information, using the surface description from the visible part could help resolve some ambiguities.

Let us consider the case where only one object is present in the scene; if more than one object is in the scene, we make hypotheses about connections to each object in the scene. We restrict ourselves to finding connections of the shadowed limbs to the existing joints only; more general procedures will clearly be needed for further extensions of this work. Shadows are likely to be caused when there are many limbs at a joint. The distance of an isolated part from all joints of the object is found, and the nearest one is picked. This joint is hypothesized to be the attachment of this isolated piece. Verification by extending the isolated piece to intersect the proposed joint is not used because of imprecise knowledge of axis direction near the end, particularly for a shadowed limb.

Hypotheses generated by proximity analysis are used when computing bilateral symmetry (Cf. sec. 5.3). If symmetry is found, we interpret it as further evidence of the

hypothesized connections. These connection hypotheses are taken as only weak evidence for linking. Thus, for the purposes of recognition, we ignore the isolated pieces first, and try to verify the connections suggested by the model (this assumes that a correct model is available). If the programs were acquiring models without human assistance, we would want to use multiple views (matching of two descriptions is discussed in sec. 6.2).

For analysis of heavily occluded scenes, where few connected structures are found initially, more sophisticated mechanisms for hypothesizing connections will be necessary. Some segmentation and line completion techniques from the work on the analysis of polyhedral objects are applicable. Edge and cross-section continuity provide evidence for connection and T-joints suggest segmentation. E.g. in fig. 4.14, the left leg of the doll is split in two parts because of the occlusion caused by the snake and hypotheses for connecting them could test for continuity (here the problem is further complicated by the touching of the snake and the lower part of the leg). We have not made any attempts at such analysis and consider it a prime problem for extensions of this research.

5.5 JOINT/PART SEMANTICS

The physical construction of a joint constrains the articulation of its parts. A ball and socket joint has different articulations than a hinge joint. The joint descriptions do not uniquely determine the physical constructions, but are suggestive. We offer some *speculations* about what object characteristics might be suggested by some joint descriptors.

Normally, we see cross-sections from one point of view, and know their width in one direction; little is known about the width in the orthogonal direction. We may assume the two widths to be the same, but sometimes we can make a better hypothesis. Consider joints where cross-section is conserved between one large piece and several, say n , small similar pieces, and the attachment point of small pieces to the joint lie approximately in a straight line. For example, consider the joint of fingers and palm of a glove. It is reasonable to guess in such cases that the cross-section of the large piece is elongated in the perceived direction by a factor of n to 1. (see Fig. 5.3(e)).

Some estimates about the invariance of joint descriptors can be made. Generally, those descriptors depending on angular relationships will change. Size related descriptors are more constant e.g. neck joints and cross-section conservation. Our attempts at estimating limb articulation characteristics from the joint descriptors derived from the observed surface have not been successful.

CHAPTER 6

RECOGNITION

Recognition of objects has been one of our major goals. It also provides a measure of the effectiveness of our description mechanisms. We regard the problem of recognition as a problem of comparing descriptions of two objects; recognition is taken to be the process of discovering whether the observed description is a description of some previously seen or known object. We believe that the most effective procedure is to verify that the observed description is compatible with the model. Guessing of a suitable subclass of models is an important and difficult part of the recognition process.

The matching procedures must take into account the following difficulties. We allow for arbitrary scale changes of an object and articulation of its limbs. Descriptions generated for an object are not entirely invariant to viewing angles and articulation. The amount of self-occlusion varies. There are missing pieces and less frequently, extraneous pieces. Only the "front" of an object is seen and the cross-sections for the parts are only partially known. Non-circular, partial cross-sections are subject to change with different viewing angles. It is necessary to be able to make partial matches when some parts of the object are invisible. It is felt, and hoped, that in spite of these variations, enough is "seen" to distinguish the scene, for most viewing angles.

For recognition, the programs need to have access to a store of model descriptions. Memory models may be constructed by storing previous machine generated descriptions, or put in by hand. Manual construction of the models may be either by making measurements of the physical object, or by supplying the description of an idealized object; however, the manual input of models is tedious. Machine generated models may use one or more views of an object. Models obtained from one view will usually be incomplete, as only parts of an object are visible; other views must be used to obtain more information. We usually construct the models by storing a description of the object from a single view; these descriptions are modified interactively to correct for errors.

Some objects can be well described in more than one way. For example, Fig. 6.1 shows a segmentation for the same glove as in Fig. 4.9 (part of the palm of the glove was not visible in the latter figure). The two descriptions have a different structure; in one description all five fingers are attached to one end of the palm, while in the other the thumb is attached to the other end. Here, we store both descriptions. During recognition, we match with each description independently and choose the one which provides a better match. In our implementation, a user makes the decision about the use of multiple, independent descriptions (the different descriptions still being generated by the machine). Automating this process is a natural "learning" problem. In sec. 6.8, we discuss how more complete models may be acquired by the machine.

Some of the object descriptors used in our system can be usefully viewed as defining a graph structure, with the joints as nodes and pieces as arcs, or vice versa. Descriptions of pieces and joints can be associated with the nodes and arcs as labels, or properties. Relations between two parts can be expressed as relation arcs; representation of relations between more than two parts is not so convenient. Our interpretation of this graph is more than just as a "syntactic" description. Many "semantic" properties are represented in it, by the choice of descriptors used for the constituent pieces, the joints and their relationships, and the use of the distinguished pieces (see ch. 5). We treat the connectivity information about an object as describing a graph structure,

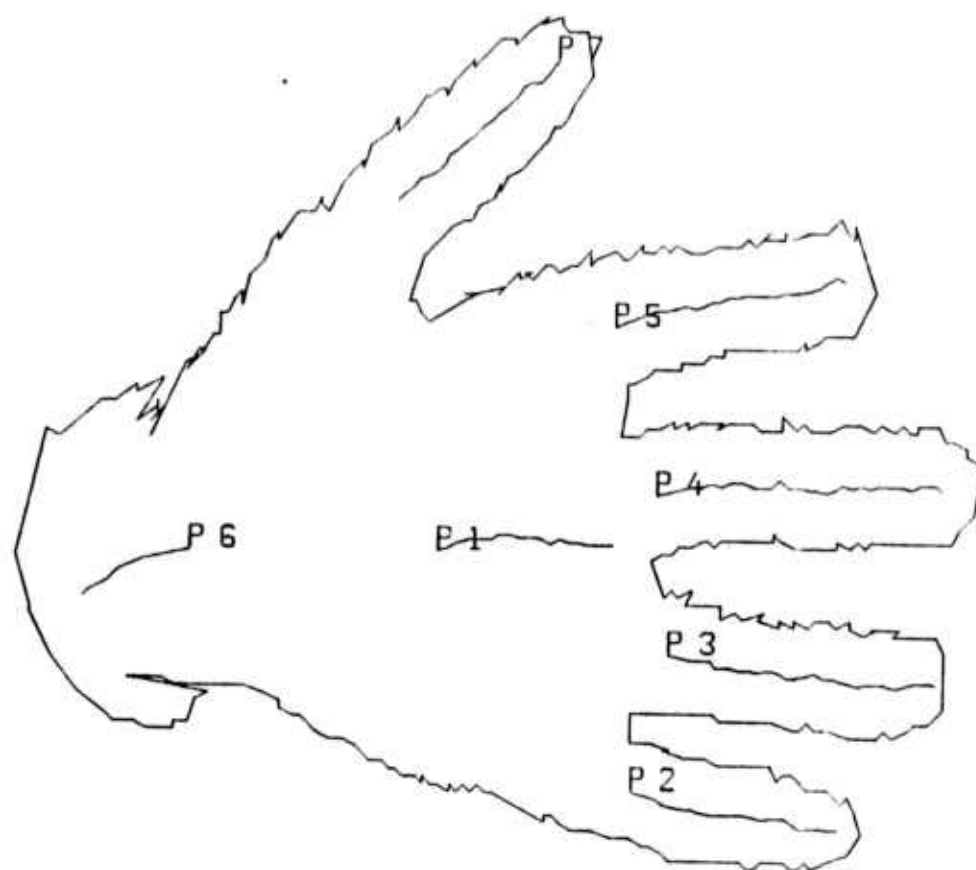


Fig. 6.1 Another View of the Glove of Fig. 4.9

however not all object descriptors are embedded in this graph form (e.g. the bilateral symmetry descriptors).

The recognition problem could be considered as a graph matching problem that finds the best partial match. Most of the general graph matching algorithms are concerned with the problem of determining complete graph isomorphisms, they provide a yes/no answer and no measure of partial match is used. They are clearly of little use for our purposes, with the expected variations in graph structures. Some partial graph matching algorithms have been proposed. A good survey of these techniques is provided in [Barrow]. Ambler et al. ([Ambler]) describe a new "maximal clique" matching technique, which finds the maximal self-consistent set of correspondences between the two graphs. We feel that the general partial graph matching procedures can be made to work for many cases, and that the graph interpretation for the descriptions is a useful one, but that treating the problem as a pure graph matching problem is a wrong emphasis, and does not lend to easy use of heuristics and scene semantics. It is hard to include knowledge about the best uses of descriptions in a graph representation alone; the graph matching procedures treat the matching of all nodes "uniformly", and use of context is difficult, e.g. we may want to match two parts (nodes) in pairs only, or insist on certain nodes being in a particular order. We will describe matching procedures, that do match the two connection graphs. However, the matching is guided by knowledge about the nature of the descriptions used.

We have attempted to make the object descriptions correspond to our intuitive descriptions; this allows us to use intuition and introspection for developing heuristics for use in matching. Humans, normally generate much more complex descriptions and use much more knowledge in their perception, however, when presented with a "stick figure" that corresponds to our machine descriptions, (e.g. see fig. 6.2), they have little trouble in identifying the objects. We can examine, introspectively, some of the processes used. Our programs rely heavily on the structure of the object and relative sizes of its parts, so do humans; the articulation angles are important for people but limited stored models prevent the programs from using them. Humans undoubtedly also use some complex mechanisms, such as an evaluation of the stability of the objects; however, such mechanisms are hard to isolate and implementation is difficult because of their complexity and lack of relevant knowledge. It is easy to provide specific knowledge about specific objects; the difficulty is in incorporating knowledge that is likely to be useful for at least a significant "micro world". Models for classes of objects, such as the class of four legged animals would be useful.

Our paradigm for recognition is as follows (Cf. the recognition block in Fig. 1.1). We use important features of the symbolic descriptions to *index* into memory models to find a subclass of similar models and compare the description of the object with the descriptions of these models (section 6.7). Each comparison generates a difference description. We pick the preferred difference descriptions, based on the similarities of structure and the similarities of the individual parts and their relations. This is the process of *direct matching* (section 6.2). *Verification* would consist of checking whether the differences between the model and the object descriptions can be "explained" in a satisfactory way, using redescription of parts if necessary. (Verification methods are discussed in section 6.6 and redescription methods in chapter 5; neither have been implemented.) If a satisfactory match is not found, new members from the visual memory can be obtained by indexing with a modified description code. We have defined different levels of the matching process: indexing, direct matching, verification and redescription.

The object description is matched with the descriptions of each model suggested by the

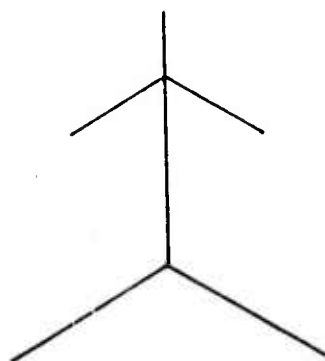
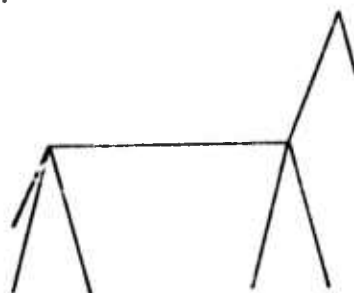


Fig. 6.2 Stick Figures of Two Objects

indexing mechanism. The result of a match is a description of the differences (and not just a numerical value). We believe that for complex descriptions a simple weighted numerical evaluation is not sufficient; passing on symbolic differences allows other procedures to use more context in making a decision. Of course, at the final decision stage, some overall assessment must be made, but at any intermediate stage, symbolic structured differences are much more useful. It helps find similarities as well as differences, such as a new object is similar to the ones we have seen before, but differs in some small respect. We think this would be essential to a "learning" scheme. Availability of explicit differences is also important, if verification and redescription is to be attempted.

Matching of the object description with a selected model description involves the matching of the two description structures and the details of the parts of the two structures. Knowledge of scene semantics guides the matching process. The matching begins by matching similar distinguished pieces. Order of pieces at two joints are preserved during matching. A match description contains pairs of joint matches, pairs of piece matches and lists of unmatched parts. With each matched pair is associated a description of the quality of that match; joint matches note the number of missing or extra pieces and piece matches note an evaluation of the match of various piece descriptors. All matched joints are required to have consistent connectivity relations.

Partial structure is sufficient, in many cases, for identification. For heavily occluded scenes, generated either by multiple objects or by the self-occlusion of a single object viewed from a particular angle, the structure of the object is not directly available; various alternate structure hypotheses must be generated and investigated. Our programs do not handle such extreme situations; these problems are further discussed in sec. 6.4.

Parts of the objects are compared on the basis of their metric properties; these descriptors are subject to some variations with the viewing angle. The length of the limbs is independent of the viewing angle; not all of this length is always seen, however. Shadow information tells us whether this is so, and such limbs can be treated differently. The visible part of the cross-sections changes, but for most cross-sections the variations are small; moreover, if the cross-sections in the model are completely determined, partial cross-sections can be matched. In a limited context of matching one part to a restricted number of other parts, these descriptors are usually distinct enough to provide adequate discrimination.

To pick a best match, we have to choose between two difference descriptions. Our decision routines attempt to choose on the basis of gross differences in the structure first and use details of the matches later, as necessary. Due to the expected variability in the descriptions of the same object, we make a choice only if the two difference descriptions are clearly different, otherwise multiple choices are reported. If alternative descriptions exist for an object (or the model), then the descriptions having the best match are selected. (Note, in the current implementation the alternative descriptions for an object are limited to merging of two separated pieces. The models may have more than one independent description.)

Further decisions require a "verification" of whether the two descriptions could reasonably represent the same object, by trying to find explanations for the differences. For example, verification may explain the occlusion of a missing piece or check the functional requirements of the model. Redescription of parts is necessary to explain some of the differences. We have not used any verification techniques; some are suggested in Sec. 6.6.

In the following, we first present an example first, and then discuss the details of matching and indexing. Appendix 2 has a concise summary of the algorithms used.

6.1 AN EXAMPLE

An example of matching a description is presented here. Some of the operations mentioned here are described in more details in the following sections. Fig. 6.3, shows the boundary for a doll and its piece segmentation, and Fig. 6.4, its connection graph. Note that one arm and one leg of the doll are not connected to the rest of the object, but the arm is hypothesized to connect to the arm joint and the leg to the leg joint. The body and the head are labelled as two *distinguished* pieces, being the two large pieces in the description.

The indexing process suggests matching this description with the stored descriptions of a doll and a horse (among the objects known to the program). Here, we will discuss matching with a doll in detail.

The connection graph for the doll model is shown in Fig. 5.1. The head and the body

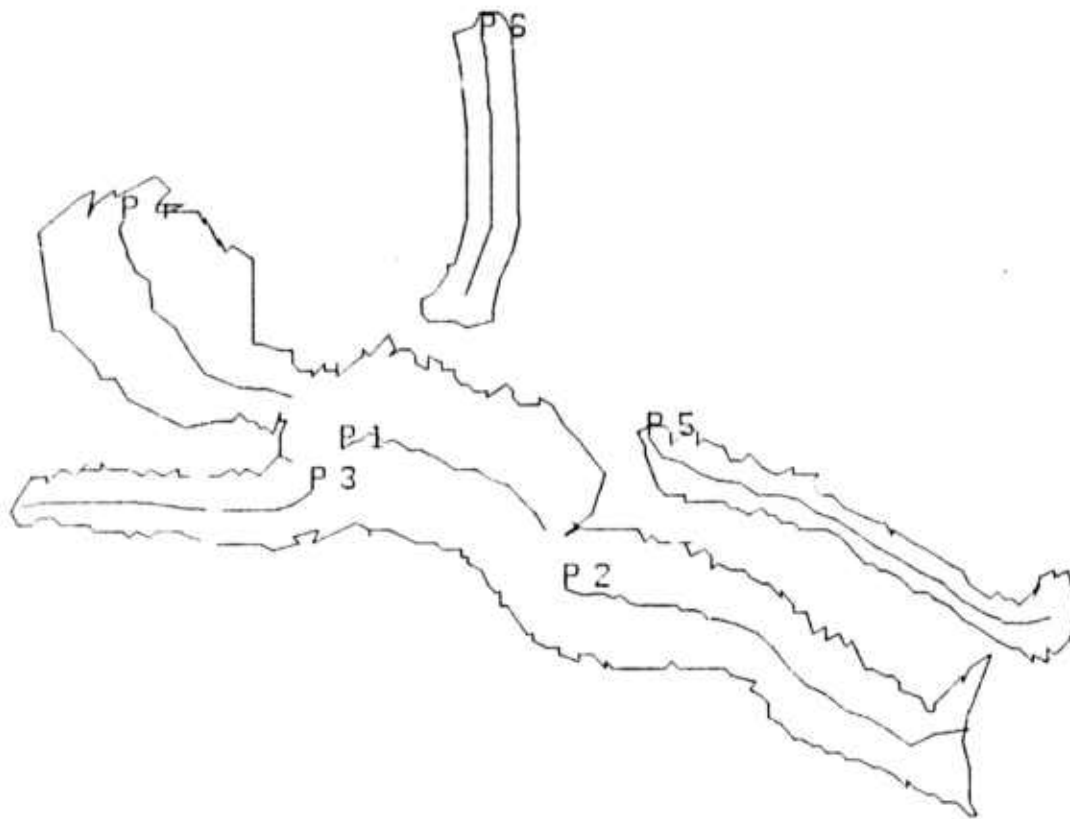


Fig. 6.3 A View of a Doll

are again the distinguished pieces. The body as a distinguished piece is two-ended (connected at both ends) in both descriptions and the head one-ended (connected at one end only). The matching starts by matching similar distinguished pieces and a two-ended object piece can not match a one-ended model piece. Thus the initial choices are:

1. object body with model body.
2. object head with model head.
3. object head with model body.

Consider the first alternative, i.e. matching the object body with the model body. The match of these two pieces is acceptable. Two choices are possible for matching the joints:

- a). the object arm joint with the model arm joint;
and the object leg joint with the model leg joint.
- or
- b). the object arm joint with the model leg joint;
and the object leg joint with the model arm joint.

The matching programs explore both alternatives. Consider option (a), and the details of matching the two arm joints.

The object doll arm joint has only two pieces besides the body attached to this joint

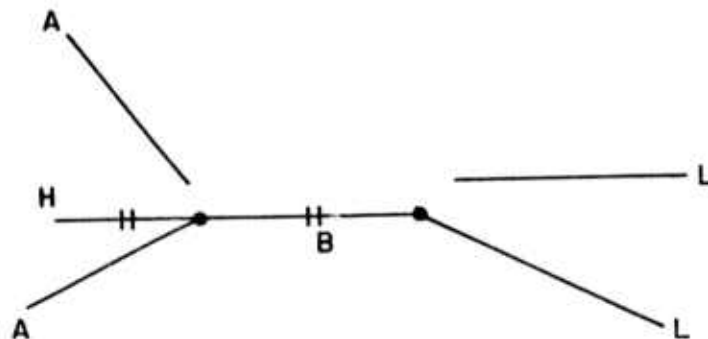


Fig. 6.4 Connection Graph of the Doll in Fig. 6.3

(head and one arm), whereas the model description has three. These two lists are matched with each other, in direct and reversed orders. For each order, the head matched with head and the arm with an arm gives the minimum total piece match error. (Piece match error is a numerical evaluation of the differences in the relative sizes of the pieces and is described in sec. 6.3). The left arm of the object matching with the left arm of the model gives a marginally better match; we have no real strong discrimination between the two orders here. Note that the information about the angle between the limbs is not used, since the model does not have any information about articulation limits. Having settled on these matches, the programs note that the model has one extra arm and the object has an isolated arm, that could be connected to the joint being matched. This match is tried and found to be satisfactory and is retained.

Matching of the two leg joints proceeds similarly. In this case the isolated object leg is shadowed and its perceived width is smaller than the corresponding leg in the model description. However, it is known that the object leg is shadowed along its width and is allowed to match with the larger model piece.

Now, examine the matching of the joints as in alternative (b) above, i.e. matching the object leg joint with the model arm joint and vice-versa. The matches obtained are: the object leg with a model arm at one end; and the object arm with a model leg, and the object head with another model leg at the other end.

A choice is made between alternatives (a) and (b) now. The average piece error is clearly better for choice (a), (the ratio is $> 2:1$). The main discrimination was made by the mismatch of head and leg for the alternative (b).

Other possible distinguished piece matches are tried. Matching the object head and the model head, ends up in a match that is identical to the above match. The other alternative of matching the object head with the model body is carried out, but turns out to be clearly inferior.

as no possible matches can be found for the object leg if the connectivity relations are to be followed. For the current example, the initial piece matches have used all the pieces and the joints of the two descriptions, and no extensions of the matches to the other joints are needed. (If the descriptions included the details of the hands at the end of the arms, we would extend the matches to the hands now.) Correct correspondence of the parts results from the matching of these two descriptions.

The object description is also matched against the model of a horse, and the following piece matches result as the best match between the two: the doll body with the horse body, the left doll leg with a rear horse leg, the right doll leg with the horse tail, the right doll arm with a front horse leg, and the doll head with the horse neck. No matches are found for the isolated doll arm, nor for one rear leg, one front leg and the head of the horse.

The connectivity relations of the matches with a doll and a horse are identical, as they should be. More parts are missing in the match with the horse, but we allow for the possibility of the parts to be hidden. The choice between the two matches is now based on the errors of piece matches. Match with a doll is preferred. However the discrimination is not strong enough to reject the other possibility. The output of the matching routines is shown in Fig. 6.5. The models that match with the description are shown in a preferred order. For match with each model are shown the assignment of the object pieces (as in fig. 6.3) with the pieces in the model (the names of these pieces are shown). The head of a doll and the neck of the horse do not match well, however the error evaluation generated by the piece match routines is not strong enough to make an unequivocal overall choice. Added discrimination requires more careful matching of the individual pieces. If articulation limits of the models were known to the models, the information about the angles between limbs would provide clear discrimination for this example.

Results for more scenes and conclusions drawn are presented in the next chapter. The following sections describe the matching processes in more detail.

6.2 MATCHING WITH A MODEL

Consider matching an object description with a particular, selected model description. In the following, we assume the model description to be just a previously encountered object description. All parts of the object are assumed to be present in the model, however, not all the details of a part are known. The full cross-sections of the parts of a model are not known, but only the perceived width from a particular viewing angle. We permit the limbs of an object to be articulated, but the present models contain no information about the limits of articulation. In section 6.8, we discuss how improved models may be acquired.

The matching problem is to make the best correspondences between the two descriptions and generate a description of the remaining differences. The problem is combinatorially difficult if all possible correspondences are tried; further the evaluation of such undirected matches is difficult. The number of alternatives considered is limited by starting the matching process by matching only similar *distinguished* pieces in the two descriptions (distinguished pieces were described in sec. 5.3).

The widths of these pieces are used to normalize the scale of the complete object and

matches in preferred order

DOLL

PRINTING PIECE CORRESPONDENCES

P3 ARM
P4 HEAD
P6 ARM
P1 BODY
P2 LEG

NO MATCH FOUND FOR THE FOLLOWING PCS OF THE OBJECT

none

NO MATCH FOUND FOR THE FOLLOWING PCS OF THE MODEL

LEG

HORSE

PRINTING PIECE CORRESPONDENCES

P1 BODY
P5 COMB_REAR_LEG
P2 TAIL
P3 FRONT_LEG
P4 NECK

NO MATCH FOUND FOR THE FOLLOWING PCS OF THE OBJECT

none

NO MATCH FOUND FOR THE FOLLOWING PCS OF THE MODEL

HEAD
REAR_LEG
FRONT_LEG

Fig. 6.5 Matching Results for the Doll of fig. 6.3

are not used for a similarity test. We require an approximate match of the elongation and cone angle descriptors. A piece may be connected either on one side only, or on both sides. Such pieces are called one-ended and two-ended respectively. A two-ended piece in the object description is not matched to a one-ended piece in the model but the reverse is allowed; we expect the object description to have no more pieces than the model. The distinguishing characteristic of the two pieces must match; if one piece is elongated so must the other.

Consider the matching of two distinguished pieces ODP (object) and MDP (model). Let the joints at the ends of ODP be OJ1 and OJ2; and at the ends of MDP be MJ1 and MJ2. These joints can be matched in two ways: 1) OJ1 with MJ1 and OJ2 with MJ2; or 2) OJ1 with MJ2 and OJ2 with MJ1. If one end of each main piece is unsymmetrical then the corresponding ends are matched. Otherwise, the choice of ends to match is based on the quality of joint matches in the two alternatives.

Now consider the matching of joints in the two descriptions, with the specified correspondence of one piece from each joint. With each joint is associated an ordered list of pieces connected to it. The order of the pieces was determined by the position along the boundary of the object. This order is not necessarily invariant with the viewing angle; however, we assume it to be so. (Note that since we are using distinguished pieces, the number of alternatives considered is suitably small that all matches without preserving order could be evaluated.) The pieces at the joints are matched in the same order and reversed order. One order is picked from the results. We are not able to differentiate between views of an object from the "front" or the "back", thus a human left hand is not distinguished from a right hand. Such distinctions normally need finer details of the surface or the cross-sections than are available to us from our hardware/software system, e.g. information about nose and eyes is useful in distinguishing the front of a human. Some improvement in resolution could be achieved by selective verification (sec. 6.6).

Matching of two pieces generates a description of their differences. The sizes of the pieces are normalized by the given scales (used throughout one complete match). We note the differences in the various piece descriptors. The descriptors used are: length, width, length to width ratio (redundant), cone angle and the number of connected pieces. We also generate a numerical evaluation based on a non-linear weighting of these differences, and call this the "error" of the match. (Details of the evaluation function are discussed in sec. 6.3.)

If one of the pieces is a complex piece, i.e. made up of a combination of pieces, then we match the whole piece as well as its components to the other piece and pick the pair that matches best. For example, our model of a horse contains two alternative descriptions of the rear legs; as a single piece or segmented in top and bottom pieces. If in some view of the horse, the whole leg is seen, it gets matched to the single leg piece in the model, but if only the top of the leg is seen it gets matched to the piece in the model describing that part of the leg.

In joint matches, the number of pieces at the two joints is not necessarily the same. We want to pick piece matches so that each piece in the smaller list is matched to one piece in the larger list (no duplicate matches) and the total match is optimal. Matching error for a list match is the sum of the errors of its component piece matches and is the criterion used for choosing between list matches. We match lists in the direct and the reversed directions and choose on the basis of resulting total errors. In this case a simple numerical evaluation suffices, since all other differences are the same. Details of the piece list matching procedure are discussed in section 6.3.

The description of differences for a joint match consists of the following: the total matching error of the associated pieces, maximum of the piece match errors, lists of unmatched (extra) pieces and missing pieces. These descriptors summarize a joint match, the piece match descriptors are also carried along with the joint match allowing the decision routines to have access to them. Note that we have not used the differences in angles between the pieces of the joints. We have no information about the allowed articulations of the limbs of a model and so assume all possible articulations. For recognition of unarticulated objects, these angles would obviously provide powerful discrimination. The descriptions of joint types have also not been used as some of them depend on limb articulations and also because of the inadequacies of low level descriptions (sec. 5.2). The nature of the matching mechanism allows easy addition of such information to the programs in the future.

After the various distinguished pieces and their end joints have been matched we attempt to choose between the matches (decision routines in section 6.5). If some matches are clearly preferable to others we retain only those. All acceptable matches are then "extended" to include the rest of the pieces and joints of both the object and the model. For each pair of pieces that have been matched, we match the joints at the unmatched ends and continue until all joints and pieces have been matched (some joints and pieces have to be matched with null joints and null pieces). This procedure assures the matching of the joints in the two descriptions to have consistent connectivity relations, if the graphs matched have no loops, as is the case for all the objects considered here. More generally, we expect the object piece connection graphs to have very few loops; our method can easily be extended by first detecting the loops and disconnecting them, then performing the extensions of the match as described and then rechecking the connectivity relations demanded by the loops. After the matches have been extended, we attempt to choose among the various matches again and the best of them is the representative match with this particular model.

Partial matching proceeds in a very natural way in the processes described here. We match those parts that are visible and make a note of the parts in the model that are not seen in the current scene. Decisions about the importance and the plausibility of the missing parts is left to the decision routines (sec. 6.5). Of course, the discrimination of the matching procedures decreases as the number of parts seen decreases.

We have used the order of pieces relative to a distinguished piece and assumed that these distinguished pieces are visible. In scenes where these distinguished pieces are hidden, we have circular lists of ordered pieces and a larger number of possible matches will need to be investigated. With the resolution of our setup, we do not normally see the details at the ends of pieces, which could be advantageously used for discriminating between pieces (e.g. the hand at the end of a human arm). Some improvement could be obtained by better procedures to examine piece terminations, in the process of the piece descriptions. In this section, we have been considering matching against a given model; however, the difficult problem with partial information is to select suitable models to match against. The problem of matching occluded scenes is further discussed in sec. 6.4.

6.3 MATCHING OF OBJECT PIECES

In this section, we discuss the details of the matching of pieces, the basis for choosing among piece matches, and the optimal matching of two lists of pieces.

Matching of two pieces involves comparing their piece descriptors. We associate a description of the differences with the piece match (a LEAP item, differences stored as datum of the item) and also generate a weighted numerical error based on the differences. The numerical error is useful for choosing in limited contexts.

The most important characteristic compared is the connectivity of the pieces to be matched. Piece matches are called as sub processes of a joint match process, and the joints at one end of the pieces to be matched are given. We compute the difference in the number of pieces attached at the other ends of these pieces, this difference is called *connectivity difference*. The connectivity difference is positive if the object piece has more pieces connected to it than the model piece, such a match will necessarily leave some pieces of the object unmatched and is thus a poor match even without further context. If the model piece has more pieces connected to it than the object piece, then the connectivity difference is set to zero; the model is allowed to have extra pieces. Only "well-defined" pieces are used for computing the connectivity differences, a piece is well-defined if it is elongated (length to width ratio larger than 3.0) or if it is a distinguished piece.

The scale of the two pieces to be matched is normalized by given factors (the sizes of the pieces first matched in the overall match). Differences in width, ratio of length to width and cone angle are computed and their weighted sum is used for a numerical evaluation of the error of the match. This error function is used only to find gross metric differences between two descriptions and the choice of the specific function is not of much importance (some reasons for the choice are explained later).

The error function is computed as follows (modifications for matching shadowed pieces are covered later):

$$\text{VALUE_OF_MATCH} = f_w(d_w) + f_r(d_r) + f_c(d_c)$$

Where:

$$\begin{aligned} d_w &= \text{Width difference} \\ &= \text{ABSOLUTE}(\text{Width}_1 - \text{Width}_2) \end{aligned}$$

$$\begin{aligned} d_r &= \text{Length to Width Ratio difference} \\ &= (\text{Length_Width_Ratio}_1) / (\text{Length_Width_Ratio}_2) \end{aligned}$$

$$\text{If } d_r < 1 \text{ then } d_r = 1/d_r$$

$$\begin{aligned} d_c &= \text{Cone angle difference} \\ &= \text{ABSOLUTE}(\text{Cone_angle}_1 - \text{Cone_angle}_2) \\ &(\text{All angles in radians}) \end{aligned}$$

$$f_w(d_w) = d_w$$

$$\begin{aligned} f_r(d_r) &= \text{If } d_r < 1.0 \text{ then } 0.0 \text{ else} \\ &\quad \text{if } d_r > 5.0 \text{ then } 1.0 \text{ else } 0.1 * (d_r - 1) \end{aligned}$$

$$f_c(d_c) = \begin{cases} \text{If } d_c < 0.1 \text{ then } 0.0 \text{ else} \\ \text{if } d_c > 0.3 \text{ then } 1 \text{ else } d_c \end{cases}$$

In the above evaluations, lower and upper thresholds have been set for cone angle differences and length to width ratio differences. If the lower threshold is not exceeded, the differences are not considered significant and do not contribute to the error value. If the upper threshold is exceeded, it indicates a very poor match and the error contribution is set to a value (1.0 in both cases) much larger than expected in a good match. Reasons for the choice of relative weights in this function are discussed later.

If an object piece is shadowed, the piece descriptors for the complete piece are unknown, and the quality of its match with a model piece is difficult to establish. We assume that an arbitrary amount of the piece may be obscured. In the above error computations, if the width of the shadowed piece is smaller than that of the model piece being matched with, the width error component ($f_w(d_w)$) is taken to be zero. Average width has been used here; if part of the piece is unshadowed then the use of the width of the unshadowed part would give improved results. The cone angle and the length to width ratio for a shadowed piece are not known reliably and are not used for determining the match error. However, the connectivity difference is computed in the same way. The information about the shadowed piece having excess pieces connected to it is still equally significant.

The selection of the error function has been ad hoc; it is based on our expectations of reliability and invariance of various descriptors. We expect the width of a piece to be known reliably (but dependent on the viewing angle); while the length, and hence the length to width ratio, and the cone angle tend to be sensitive to description methods. An improvement would be to assign the weights for each piece match separately, depending on some context e.g. for matching with a very long piece the elongation is important and should be given more weight. A more complete model of the objects might specify what the essential qualities required for a piece are and influence the weighting. Standard, statistical parameter setting techniques ([Duda]) may be useful in determining these weights, if no context is used.

The shapes of the piece axes (straight, circular etc.) have not been used for matching, primarily because the pieces encountered in the objects we consider have been mostly straight. The cross-section shapes have not been matched, since these are not known very well in the present implementation. We have also not compared the cross-section functions in a very detailed way; we merely fit a straight line to one, to determine an average cone angle. A more subtle evaluation could compare the individual cross-sections along the axes of the two pieces and build differences; this is expected to catch local differences of shape better than our averaging process. The major difficulty would be caused because of the quality of the boundary data, which adds a significant error to the cross-section widths, masking any fine systematic differences.

To select between two piece matches, we first compare the connectivity differences. If one match has a higher connectivity difference (which corresponds to excess object pieces) than the other, then that match is rejected. If connectivity difference is the same, we pick the piece with the lower numerical evaluation. This piece selection method is very local and is used in limited contexts only (in selecting piece matches when list of pieces at two joints have to be matched).

MATCHING OF PIECE LISTS:

Consider the matching of two lists of pieces, in a given order; the lists may have different number of pieces. Let the two lists be $P = \{p_1, p_2, \dots, p_m\}$ and $Q = \{q_1, q_2, \dots, q_n\}$, and let $m \leq n$. We want to pick piece matches so that each piece in the shorter list, P , is matched to one piece in the longer list, Q . One piece can match with only one other piece. The relative order of pieces must be maintained, i.e. if p_i matches q_j , and p_{i+1} matches q_k then $j < k$. This requirement constrains the matching of a piece, p_i in P , to be matched to q_j in Q such that $i - (n - m) \leq j \leq i + (n - m)$. Each p_i must be matched to a different q_j . A complete tree search with these constraints could be made to find the best assignments, as the number of pieces involved is small (say ≤ 5). However, we use the following procedure which is faster: Evaluate the match of each piece p_i in P to each potential matching piece q_j in Q (with the noted constraints between i and j) and arrange the matches in a list ordered by the preference of these piece matches (piece match selection was discussed earlier). For each piece compute a match sensitivity ratio (to be defined shortly), indicating how fast the match quality for this piece deteriorates as its matching piece is changed. Assign the piece with the highest sensitivity ratio its best match. Remove the matched pieces from further consideration, by removing matches involving these pieces from the piece match lists, and update the sensitivity ratios. Repeat this until all pieces in P have been matched.

Let us now specify the match sensitivity ratio. Some pieces are left with only a single match, either initially or after some piece assignments have been fixed; in such cases, we set the sensitivity ratio to an arbitrary high value, MAXRAT, assuring that this piece will be assigned a match first. Normally, the sensitivity ratio is the ratio of the second best match error to the best match error. However, if the object piece is shadowed, the sensitivity ratio is set to its lowest value. The match quality of a shadowed piece is not known reliably, and its match selection is deferred to be last. If the connectivity difference of the two alternatives is different, the sensitivity ratio is set to $0.1 \times \text{MAXRAT}$, so that this piece will be assigned its best match immediately after the single match pieces have been matched (this applies for a shadowed piece match as well). If the piece is a "well defined" piece, the ratio is doubled (so that the well defined pieces are matched earlier).

A piece list match is characterized by the sum of individual piece match errors. To choose between two orders of matching two lists (direct and reversed), we compare the total errors for the two orders; if one order is clearly better (error ratio > 2) then pick the lower error match, otherwise make a decision based on the lower maximum piece error. A numerical piece error suffices for selection here, since the rest of context is the same for the two orders.

Many objects we consider are bilaterally symmetric; their limbs form symmetric pairs. For such objects, the matching of piece lists could be improved by matching the pairs simultaneously; the above described procedure finds matches for each limb separately.

6.4 OCCLUSION AND SHADOWS

Parts of an object may be occluded by other parts of the same object, or by the parts of another object in a multi-object scene. In such cases, the description procedures provide some isolated sub-structures, each such sub-structure consisting of a number of connected pieces, and some isolated single pieces. We regard each sub-structure as a separate object. The isolated pieces

may be parts of one of these objects or be single piece objects. Description routines also generate hypotheses for connections of these isolated pieces to the joints of objects, based primarily on proximity and symmetry (Sec. 5.4). At the stage of matching, we are able to further examine the validity of these hypotheses, by examining whether such pieces are present in the model but not in the object description. We discuss the handling of these isolated parts assuming that connected parts of the object provide enough data to make partial matches.

The matching begins by comparing a connected sub-structure, regarded as an object, with memory models, ignoring the isolated pieces that possibly attach to this object. At each joint match between the object and the model, we examine whether the model joint has some pieces that have not been matched. If so, the pieces hypothesized to connect to the current object joint, if any, are matched with these extra model pieces. Since the connectivity hypotheses are weak, such matches are accepted only if the pieces match well. (Matching of a shadowed piece with a model piece was discussed in sec. 6.3.) The matching error is required to be less than the maximum matching error of the other pieces at this joint. A more satisfactory resolution would be to use more sophisticated matching techniques, such as to determine whether the visible part of the shadowed limb matches with some part of the model limb. If a suitable match is found, then this isolated piece is regarded as being attached to the proposed joint (in the context of hypothesizing this object to be the model object), and is hereafter included in the evaluations for quality of the overall match. This process is repeated for each proposed object in the scene. (A converse procedure, that assumes the hypothesized connections first, and verifies them by matching with memory models, has the advantages of starting the matching with a larger, more selective structure, and will be particularly useful for heavily occluded scenes where little sub-structure is immediately available. A combination of both methods is likely to be used for difficult scenes.)

Isolated pieces that remain unmatched by the described procedure are matched with single piece objects. Further treatment of the pieces that remain unmatched is difficult. The current programs simply ignore them and identify the remaining objects. Several alternatives are possible: after an object has been identified and some pieces are missing, we may accept even poor matches for these pieces now. Redescription of these pieces and a closer examination of shadows will perhaps be necessary for a better treatment; we have not investigated this. Some parts are split in two because of occlusion from another part, e.g. one leg of the doll in Fig. 4.14 is seen as two separate parts because of the occlusion caused by the snake. Continuity of such parts could now be examined with the knowledge of the corresponding model piece (such as the length of the model piece is close to the length of the combined piece). This technique has not been implemented.

The foregoing presumes that enough parts of an object are seen as connected to establish a good match with the models; this may not be so for heavily occluded scenes. Some conjectures about such conditions are offered here. In such instances the matching process and the description process must work more closely together, the matching process supplying more information for description hypotheses. Some hypotheses for connections and continuations of parts can be generated at the description level, "bottom up", by examining continuity. We can use the continuity of surface, axes of parts and the cross-sections. Three-dimensional position information will greatly aid in the determination of these continuities. Alternatively, given a model description to match against, we can attempt to find parts of the pieces that fit well with the model ("top down approach"). A combination of the two methods will probably be used.

It is our feeling that the major problem is to select a suitable set of likely models based on the initial descriptions for these occluded scenes, to keep the combinatorics within control. Unfortunately, the resolution of our system does not permit us to see the details of the ends of parts, which could be used to suggest likely models. More refined piece descriptors would also be helpful here. Finally, we think that global context must be used to aid the selection of models here, i.e. we must have some idea of what we are looking for, when we see a complicated occluded scene.

An example with shadowed pieces was discussed in sec. 6.1. Some results for scenes with multiple objects are presented in chapter 7.

6.5 SELECTION OF A MATCH

Recognition requires a selection between two object matches. The selection procedures are the same, whether all the joints of the object have been matched or not. These procedures can be called at any stage of the matching process to determine whether a preferred match exists, allowing incorporation of matching "strategies". For example, our programs attempt to select first when only the initial two pieces and their joints are matched and later when the whole object is matched.

The two descriptions of the same object are not expected to be identical, and so we must have a way of choosing between two non-empty difference descriptions. In general, the problem of deciding what differences are more important than the others is difficult. One solution for this problem is to try and find unacceptable differences; descriptions are effective to the extent that such large differences can be found between most object descriptions. Our preference scheme first attempts to find major differences in the structure of the two descriptions, if large differences are found, further evaluation is not necessary. Finer distinctions, between similar structures, are made on the basis of individual part matches. If increased resolution were available, better discrimination could be obtained by examining details, e.g. by examination of the ends of legs and hands of animal shapes. The choice of preferences is heuristic, and is expected to apply to a wide class of objects. Possibilities for other preferences to be specified in the models of the objects will be discussed. Also, it would be easy for the decision preferences to be determined by the calling programs in our case, because of the availability of structured symbolic differences at the decision stage.

Each object match description contains a list of joint matches and piece matches. Some local selection decisions about what pieces to match have already been made. Global choice is attempted here. The selection is based on finding large differences, otherwise no choice is made.

The main structural difference looked for is the number of pieces in the object that are not matched by the model (consistency of the connectivity relations is already assured by the matching procedures). We are assuming the models to be complete and the excess pieces are considered to be a major discrepancy. We choose the match with fewer excess pieces. For these evaluations only well defined pieces are considered (elongated pieces only). This helps make the selection insensitive to small extraneous pieces, that may sometimes be generated during the description process. The sum of the connectivity differences (sec. 6.3) gives the number of object pieces that are not going to be matched by the model and is useful even when all the joint

matches have not been completed. We completely ignore any missing pieces, presuming them to be hidden. A more sophisticated process should investigate whether this is possible. (Simply preferring the match with fewer missing pieces will always result in picking the simpler of the two models that share a common structure, e.g. the structure of a horse is similar to that of a doll with an extra limb for tail, and this preference scheme will tend to pick a doll, for those views of the horse in which the tail is not visible. It is difficult to combine the number of missing pieces with the other measures of match quality.)

If no structural differences are found, we compare the evaluations of the piece matches of the two object matches. For each object match the average and the maximum piece match error is computed. Average error is an indication of the overall fit of the various pieces and the maximum error is useful when two objects are similar but differ markedly in a single piece. If either of these quantities differs significantly (by a ratio of 2 to 1) between the two matches, we select the match with the lower error. Average error is considered before the maximum error.

If no clear differences are obtained, we pick the match with the lowest average error or the lowest maximum error, depending on which provides the greater discrimination (larger ratio), but mark the selection as low confidence.

The decision procedure described here relies on general considerations and gross differences. We do not make any decisions based on model specific information. Part of the reason is in our use of loose models. The models are just previous descriptions with minor modifications. More specific information could be added to the models either by hand or by a description learning scheme ([Winston]). If the models specified the necessity of certain relations to hold, we could check for them here. These relations can be of the form of certain ratios of sizes of the limbs, necessary similarity (or dissimilarity) of the limbs etc. No angle information has been used in making decisions, since the articulation limits are unknown. If added resolution were available, we could examine the confusing parts in more detail, e.g. examine the structure at the ends of the parts.

6.6 VERIFICATION

Procedures to choose between two matches were described in the last section. Additionally, it is desirable to test further the adequacy of a match. Our decision procedures insist on a minimal quality of the match (such as no extra well defined pieces) but no attempt is made to explain the remaining differences. Since there is likely to be more information in a model than in the object description, a more general system should try to "verify" model information in the object description. This seems to be the case in human perception ([Posner]). Redescription of some parts may explain some of the differences; the description of model pieces is known and the description procedures can test whether a similar description can be obtained for the object pieces.

We have not implemented verification and redescription procedures. In the following we suggest some verification techniques.

1. Examine the missing and extra pieces: Can the invisible piece be hidden for some permissible articulation of this piece and the known articulations of the visible pieces? Can the extraneous pieces be included in the other pieces by redescription?

2. Can the metric differences of two matched pieces be explained? The perceived width of a piece changes with the viewing angle; examine whether the width is consistent with the present viewing angle. (For such verification, the model needs to have information about the complete cross-sections.) Sometimes, a piece is terminated prematurely because of a local discontinuity in the boundary and its length measurement is faulty, now we can redescribe this piece with different continuity conditions to determine whether it can be matched better with the model piece.

3. Model specific data: check any specific relations that must hold for this particular object (among the visible parts). These relations may be based on the functional requirements of the object.

4. Support and stability relations: check whether the object could be stable with the proposed piece assignments. This is difficult with only partial information about the object.

5. Increase Resolution: Our system is limited in resolution; however, if higher resolution were available, we might not want to process the whole scene at this higher resolution. After matching, we have specific high resolution features that we need to verify. This may involve gathering new, high resolution data from the scene, or just to make use of such data in parts of the description phase. (Increased resolution will require the use of a narrower light beam and finer image sampling; increasing the effective stereo angle is not practical because of additional shadow problems.)

6.7 INDEXING INTO VISUAL MEMORY

In this chapter, we have discussed the matching of an object description with a given model description, and also the choice between two such matches. If the number of models known is small, for recognition we can simply match the current object description with each known model and choose the best match. However, as the number of models increases, the computation required increases proportionately, and *indexing* to locate a sub-class of similar models becomes necessary. In our system, we have experimented with a small number of models only, but the number of models can be increased indefinitely, in principle. In the following, we describe some preliminary efforts at indexing and also discuss how more powerful indexing methods may be implemented.

For indexing, some important "features" are abstracted from a complete object description. The features may be viewed as forming a "feature vector" or a "description code". Note that two models may have the same feature vector and differ in the detailed descriptions. These features are used only to locate promising similar descriptions, and not to establish a detailed match. Models with exactly the same description code can be located efficiently by standard *hash coding* techniques. We have chosen those features of an object description that are insensitive to changes in the viewing angles and limb articulations. However, some variations do occur because of occlusion and description accuracy. We do not expect to find a memory model with the same description code, but instead look for those models whose description code is close to the observed code.

The problem of finding a similar code is similar to the problem of finding a best match

(with Hamming distance) discussed by Minsky & Papert ([Minsky], pp. 222 - 225). They conjecture that the solution of this problem requires the equivalent of a complete search of the memory. Rivest ([Rivest]) presents a statistical analysis claiming that for reasonable distribution of the codes in the description space, a simple search around the observed code has a high probability of finding a match in a rather small number of steps. However, these methods have treated the problem of searching for best match as a general combinatorial search problem. For our problem of indexing, the generated descriptions have some *semantic* content to them, which can be used advantageously for searching. We can use the knowledge of the descriptions to decide which descriptors are likely to be insensitive to change and also which ones may be in error for this particular description.

Our paradigm for indexing is as follows. One or more description codes are generated for an object description (also an object may have multiple descriptions). The models with the same description code are retrieved from the memory. Based on the knowledge of descriptors and possible errors, the description code is changed and new models with the modified code are retrieved. The number of changes made to the description code may depend on the confidence of the various components and the process could be stopped if a suitable match was found. However, because of the difficulties of judging the adequacy of a match, we have not chosen to stop the indexing process until all reasonable alternatives have been tried.

The choice of features used for indexing has been based on their invariance. For the class of scenes considered in this thesis, we have been assuming that one or more of the *distinguished* pieces of an object is present in any scene. This has lead to a choice of descriptors of the distinguished pieces for indexing. Each distinguished piece generates a separate code for indexing, and the presence of any distinguished piece in the scene is sufficient for proper indexing, i.e. indexing is possible from partial views. The choice of descriptors used is further constrained by the desire to use only those descriptors that can be represented by integer values, preferably binary. Use of real valued descriptors such as the relative widths of the pieces at the end is more difficult. A possible approach is to quantize the real values; however we have not used such descriptors,

Following is a list of descriptors for a distinguished piece that are usable for indexing:

1. Connectivity of the distinguished piece (connected at one end or both).
2. Conical or cylindrical distinguished piece (conical being defined by the average cone angle exceeding a threshold).
3. The type of the distinguished piece, e.g. long or wide.
4. Shape of its cross-section; flat or curved, concave or convex.
5. Shape of the axis; is it straight ?
6. Regularity of this piece (cross-section function of a regular part has some simple geometrical shape)

We have used only the first three of these descriptors, primarily because the programs for generating the other descriptors have not been implemented (see sec. 5.1 for choice of piece

descriptors). Also, the computations of regularity cannot be reliably made with the quality of data currently available to our programs. We have used the descriptors of the distinguished pieces only. However some descriptors for the whole object may be usefully included in the description code, e.g. whether any of the pieces is regular. We have not used any joint descriptors for indexing here, because of their variability as discussed before (sec. 5.2).

Further efficiency in retrieval of the models is gained by using the number of pieces attached at the either ends of a distinguished piece. Let N_1 be the number of attached pieces at one end and N_2 at the other, and further let $N_2 \leq N_1$. The models with the same description code are stored in a list, which is retrieved during indexing. This list is ordered in a descending order with the number N_1 corresponding to the particular distinguished piece. During retrieval, we search along this list for those models that have more attached pieces than the observed description piece. The models must have at least as many pieces as the object for an acceptable match. When the first model with less pieces is encountered, the rest of the list need not be considered. Further improvement would result in ordering the sub-list of models with the same value for N_1 by the value of N_2 . On the average, these two orderings should reduce the number of models to be considered by a factor of two each.

In the current implementation, we index into memory for a description code corresponding to each distinguished piece. If the object piece is one-ended, it can match with a model with a two ended distinguished piece and this alternative is also used to index. Other perturbations to the description code could be based on the confidence of the descriptors, e.g. if in the description of the distinguished piece the observed cone angle is close to the threshold then we should try its description, both as conical and non-conical; we have not used this. The number of entries required grows exponentially with the number of descriptors that need to be perturbed, and must be chosen carefully. Our experiments with indexing did not advance enough to study this in detail. In the following we present an analysis of expected indexing efficiencies, based on some simplifying assumptions.

The retrieval efficiency of the indexing scheme is dependent on the number of descriptors used. Let n be the total number of descriptors and assume all descriptors to be binary valued. Also assume that on the average m of these descriptors have value 1. If this number is assumed to be exactly m , the number of possible codes is the binomial coefficient $C_{n,m}$. Consider the situation where the number of models in the memory is much larger than the number of possible model codes. In this case each model code is expected to have a long list of models attached to it and each probe into memory is expected to succeed in retrieving some models. Let l be the number of descriptors that are doubtful and need to be perturbed. Then, the number of entries made is 2^l (assuming independence of these descriptors). The reduction in the number of models considered in this case is the number of possible codes, divided by the number of entries made, i.e. $(C_{n,m})/(2^l)$. A further improvement by a factor of 4 can be expected by ordering the list as described above.

As example, if n were 6 (as enumerated in the list of descriptors above) and m was 3, $C_{n,m} = 10$. The best possible improvement factor is then 40, modified by the number of needed entries. For our implementation with $n=3$, and $m=1$ or 2, the expected improvement is still equal to 12. Considering the preliminary nature of these efforts, we feel that the results are encouraging. Note that our expectation of indexing efficiency is predicated on the belief that the objects will be evenly distributed over the chosen descriptors.

Consider the example of the doll discussed in section 6.1 (Fig. 6.3). The set of models used consists of: a horse, a doll, a glove, a ring, a snake and a hammer. The distinguished pieces for the present view of the doll are the body and the head. Using descriptors of the body, the indexing programs pick out the doll and the horse as the similar models (the other models do not have enough attached parts to their distinguished pieces). However, using the head of the doll, the hammer model is also picked out as a likely model. Note that if the shadowed arms were connected to the body, this suggestion would not have been made. The hammer is quickly rejected as a possible match, by comparing the *total* number of pieces of the object. This number could also have been used as a descriptor for indexing by further ordering the model lists by this number.

Current implementation of indexing suffers from the inadequacies of our descriptive techniques, though we think that some improvements can be made by techniques already discussed (see chapter 7, for a summary). Use of real valued descriptors would also aid in improving the effectiveness of indexing, e.g. we could use the relative sizes of the pieces and the angles between them. Our indexing scheme is designed to work with occluded scenes and partial views. However, situations with heavy occlusion are not considered. With a limited number of models, the analysis of occluded scenes is somewhat simpler, as various models can be "fit" to the observed data. With a large number of models, it becomes necessary to generate enough "bottom up" descriptions, so that a list of likely models can be efficiently indexed.

6.8 MODEL ACQUISITION AND LEARNING

For recognition, we need a collection of model descriptions. These models may be previously seen descriptions (visual memory) or be input by hand. The latter alternative is tedious and not used. We construct models by saving a previous description of an object, which is then interactively modified to correct for errors. A suitable viewing angle is chosen so that a maximum of the object is seen. Additional information, as necessary, is added to the model so that all parts of the object are present, but not all details of the parts are known. The full cross-sections of the parts of a model can not be determined from a single view. We store only the perceived width from the particular viewing angle. The observed angles between the pieces at a joint are known, but not their articulation characteristics. In some cases, we use independent descriptions generated by two different views, as in the example of a glove discussed earlier. Such models have been satisfactory for our purpose; following are suggestions on how more complete models may be acquired automatically.

Information about the unseen parts of an object can be obtained by using multiple views of the object. To combine the information from several views, we have to be able to find common links in various views. If the object stays in the same physical position for the different views, or is moved by a precisely known amount (e.g. by being rotated by a known angle on a turntable), then the linking problem is simpler, since we know the three-dimensional positions of points on the visible object surface. We can assemble data from the separate views before generating any symbolic descriptions, but must "register" the various views whose absolute calibration may be in error. Alternatively, we can describe each scene, and then match the descriptions of the various views. The different views can, for example, be used to complete the cross-sections for parts. We can use the knowledge about the limbs being in fixed positions here. Alternately, if many views are used, the differences from view to view can be made arbitrarily small and the correspondence problem becomes trivial.

Views with different amounts of limb articulations can be used to learn the articulation characteristics. The matching procedures can be the ones we have presented, with the exception of not assuming that either of the descriptions to be compared is more complete than the other. Since the descriptions will not be "perfect", the matching procedures must decide about which parts of the descriptions are acceptable, based on the compatibility of the different views. Parts found in several views are clearly more credible. The learning examples need not use views with heavy amounts of occlusion, making the task of description and matching easier. When two very different descriptions are generated for the same object (e.g. glove in Figs. 4.9 and 6.1), the programs will need to decide whether both of them are acceptable or whether one is an erroneous description. This decision may depend on whether one description can be transformed into the other, without changing the shape of the object. (An alternative to storing both descriptions would be to have the matching programs attempt such a transformation.) Acquisition of such knowledge is "learning" in a non-trivial sense. We think that the present matching procedures can be easily extended to accomplish this.

A different class of model characteristics, requiring certain relations to hold for a particular model, can be learned by examination of different "examples", as suggested by [Winston]. These relations may, for example, be required metric relations of some parts or required symmetry of parts. Winston's procedures find similarities and differences between different examples and abstract necessary relations for a certain model. We are able to generate similar difference descriptions. However, Winston relies heavily on each description being perfect (no missing or extra lines in the descriptions) and extension to imprecise descriptions will require addition of preference criteria.

We have not investigated these learning problems, and suggest them as important problems for further research.

CHAPTER 7

RESULTS AND CONCLUSIONS

Some results for different techniques described were presented earlier. Here we present more results on other scenes to illustrate their performance range. We summarize the strong and the weak points of our techniques and implementation, and discuss some ideas for related, further research.

7.1 RESULTS

Figs. 7.1 thru 7.15, show results at various levels of processing. These include the results of boundary routines, preliminary segmentations and axis, cross-section descriptions, selected piece segmentations and the output of the recognition routines. The segmented scenes show the axis of the cones. The matching results show the models selected by the indexing and a preferred ordering of these models as a result of matching (one or more models may be included). For each such match, the figures also show the correspondences made between the pieces of the object and the pieces of the model. In the following we discuss in detail the performance characteristics of the various processes, by examining the results.

Use of three-dimensional data is very effective in separating occluded bodies; the separation is a natural outcome of the boundary organization process (see Figs 7.3 and 7.5; compare with the TV pictures in Figs. 7.1 and 7.2). However, touching objects are not necessarily separated, e.g. part of a doll leg and the snake in Fig. 4.14 are seen as a single object.

The quality of the boundary output is affected by the following factors. The end points of the two series of laser scans do not always match well causing the boundary to be jagged. A particularly noticeable example is in Fig. 7.6. The thinning process is also poor in location of the end points of short segments. More serious problems can occur if the scan data itself is poor. If the hue of the object is complementary to that of the illuminating light (or it has dark spots), the TV image of the scan has false discontinuities. Use of a white light or multi-color laser would solve this problem in many cases. Reflections from the object can give rise to spurious image points. Combining TV image data with the laser scan data should help with the above problems. The boundary definition is of course limited by the resolution of the apparatus.

The segmentations for an object are chosen from several alternatives. Previously, in figs 4.6 and 4.13 we presented the alternative cones for two scenes. Figures in this chapter show only the selected cones. Choosing among alternate descriptions involves computation of the overlap of two descriptions. We have used boundary overlap as the measure of piece overlap, area overlap is more robust and closer to the desired measure. Area overlaps can be computed without substantial overhead because of the nature of the data (the areas are described by an axis and normal cross-sections). Proper resolution of some alternatives requires redescription techniques and was discussed in sec. 4.5.

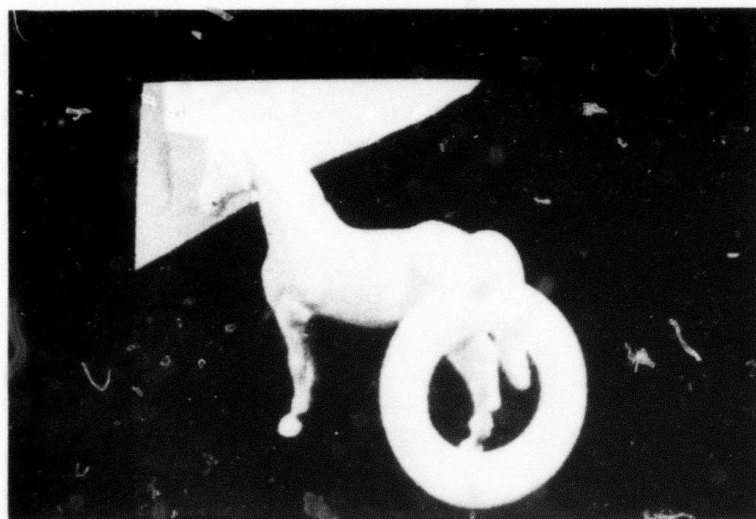


Fig. 7.1 A TV Picture of a Horse and a Ring

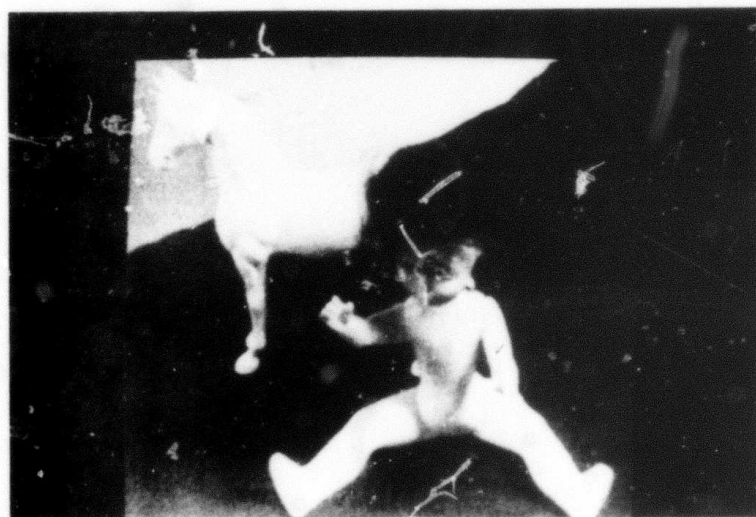


Fig. 7.2 A TV Picture of a Horse and a Doll

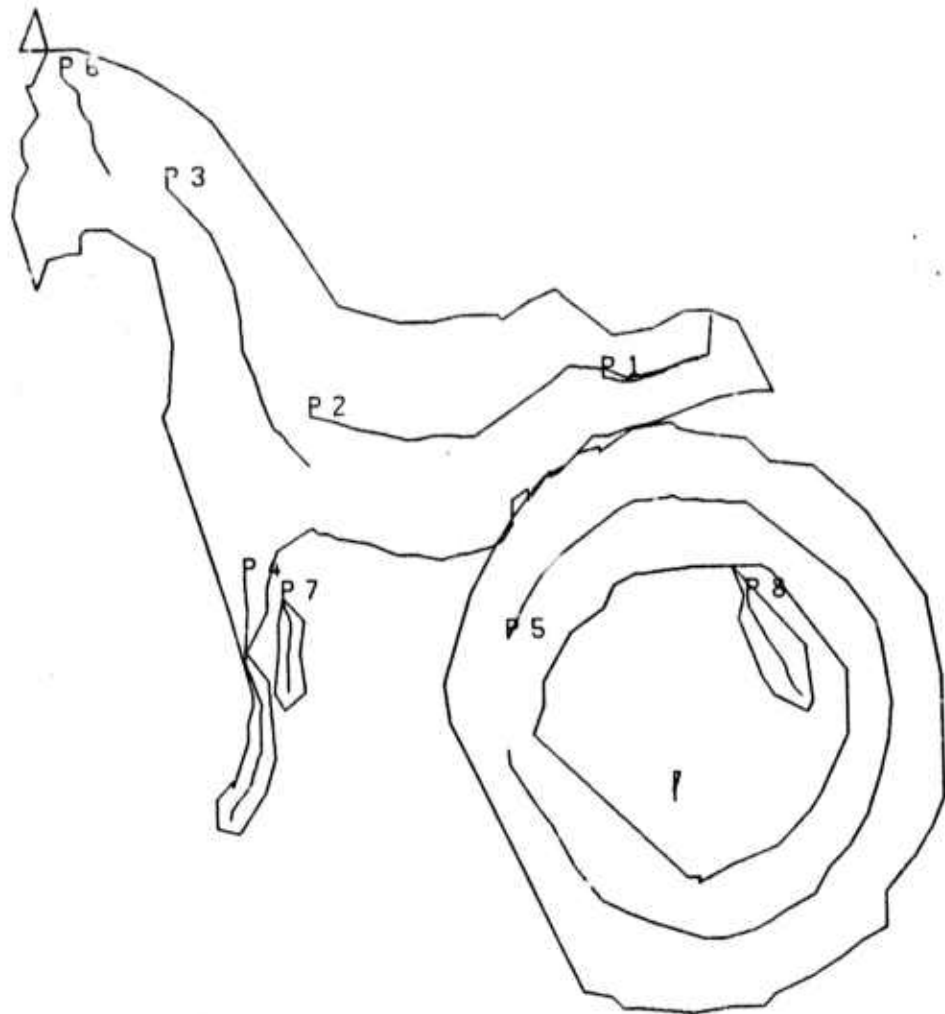


Fig. 7.3 Segmentation of Scene Corresponding to Fig. 7.1

THE MODELS SELECTED BY INDEXING:

HORSE
DOLL
GLOVE

matches in preferred order

HORSE

PRINTING PIECE CORRESPONDENCES

P9 NECK (Note P9 is the piece formed by merging P3 and P6)
P2 BODY
P4 FRONT_LEG
P1 TAIL

NO MATCH FOUND FOR THE FOLLOWING PCS OF THE OBJECT

none

NO MATCH FOUND FOR THE FOLLOWING PCS OF THE MODEL

HEAD
REAR_LEG
FRONT_LEG
COMB_REAR_LEG

Fig. 7.4 Recognition Results for Horse in Fig. 7.3

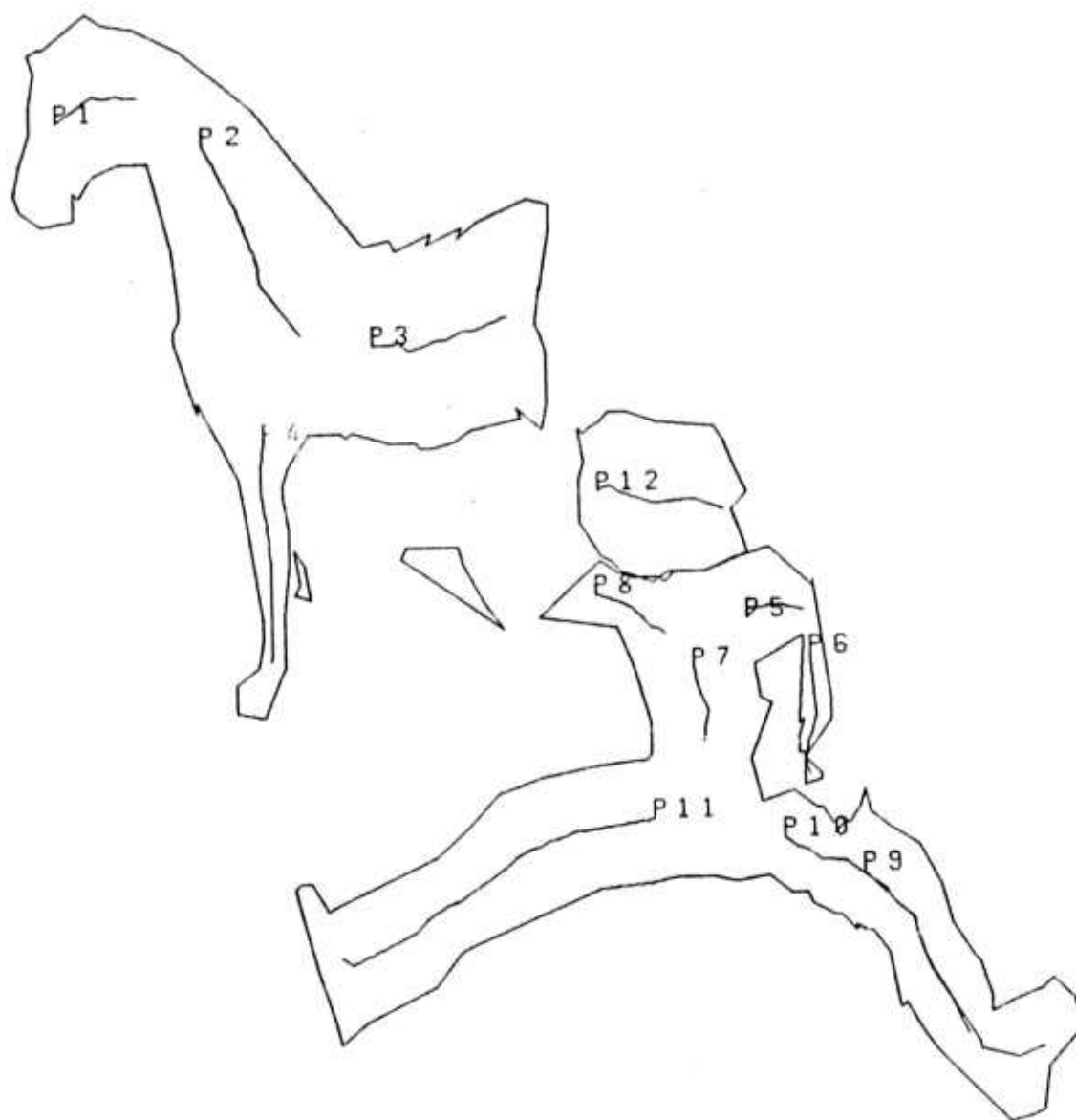


Fig. 7.5 Segmentation of Scene Corresponding to Fig. 7.2

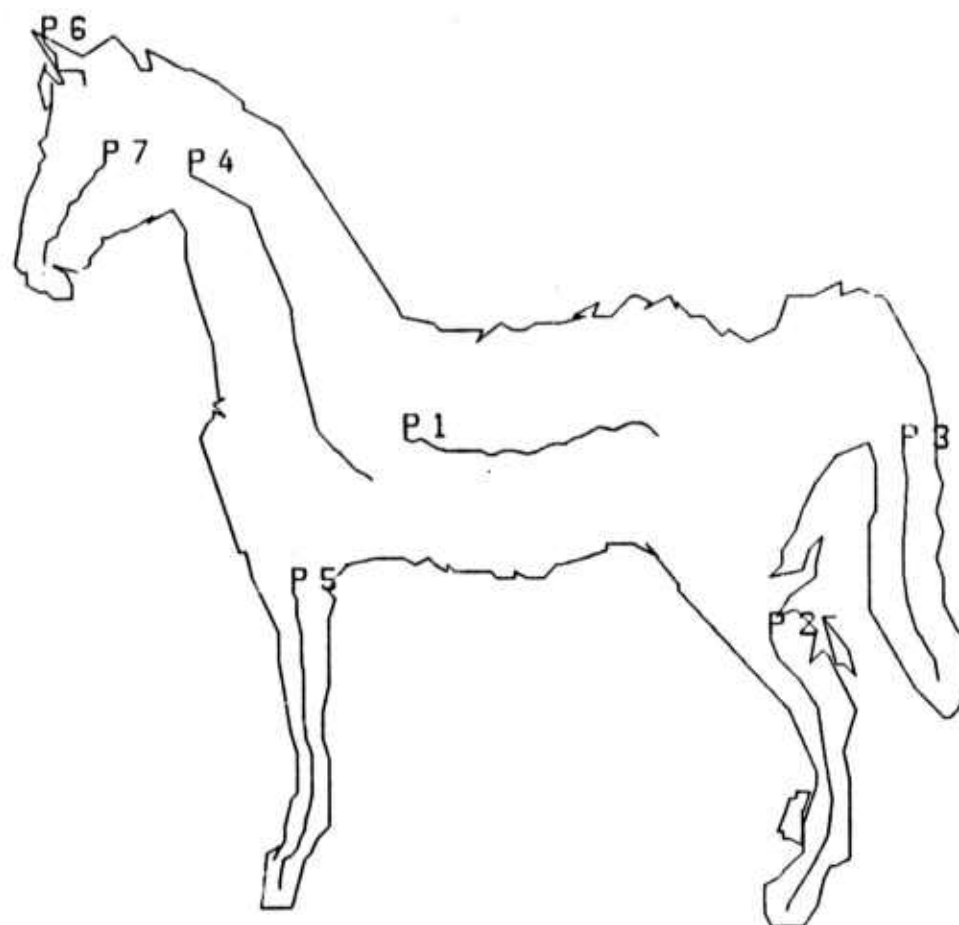


Fig. 7.6 A View of a Horse

THE MODELS SELECTED BY INDEXING:

HORSE
DOLL
HAMMER

matches in preferred order

HORSE

PRINTING PIECE CORRESPONDENCES

P1 BODY
P2 TAIL
P3 TOP_REAR_LEG
P5 FRONT_LEG
P4 NECK
P7 HEAD

NO MATCH FOUND FOR THE FOLLOWING PCS OF THE OBJECT
P6

NO MATCH FOUND FOR THE FOLLOWING PCS OF THE MODEL
BOTTOM_REAR_LEG
REAR_LEG
FRONT_LEG

DOLL

PRINTING PIECE CORRESPONDENCES

P1 BODY
P2 LEG
P3 LEG
P5 ARM
P4 HEAD

NO MATCH FOUND FOR THE FOLLOWING PCS OF THE OBJECT
P6
P7

NO MATCH FOUND FOR THE FOLLOWING PCS OF THE MODEL
ARM

Fig. 7.7 Recognition Results for Horse of Fig. 7.6

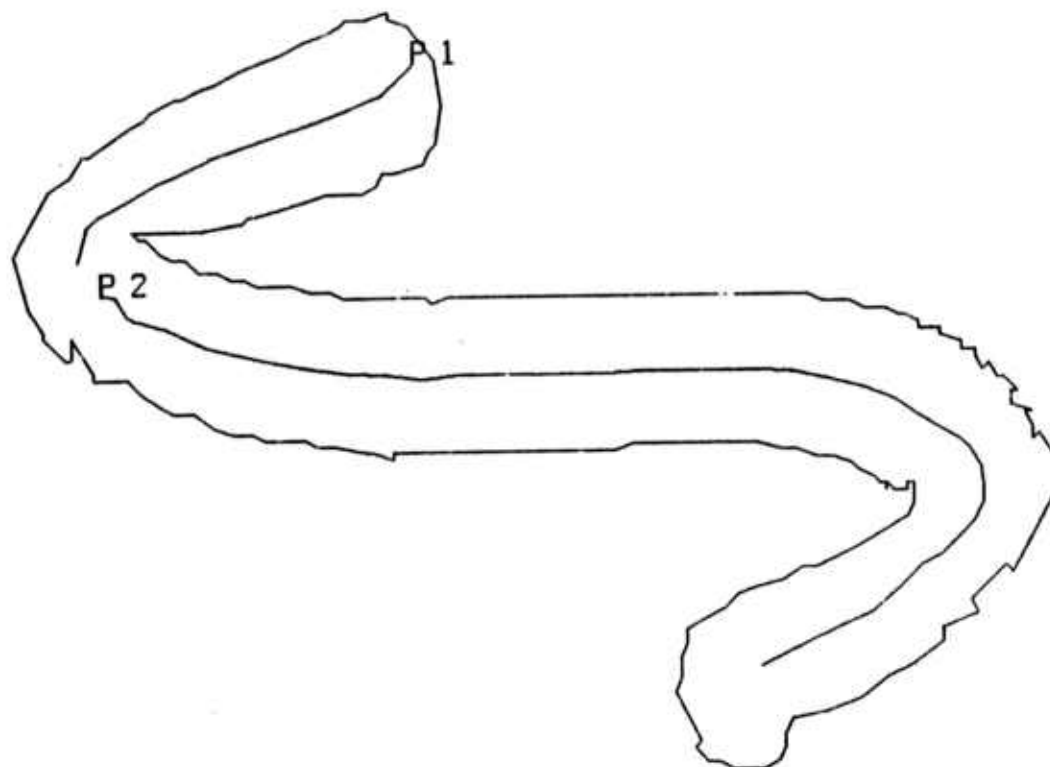


Fig. 7.8 A Snake

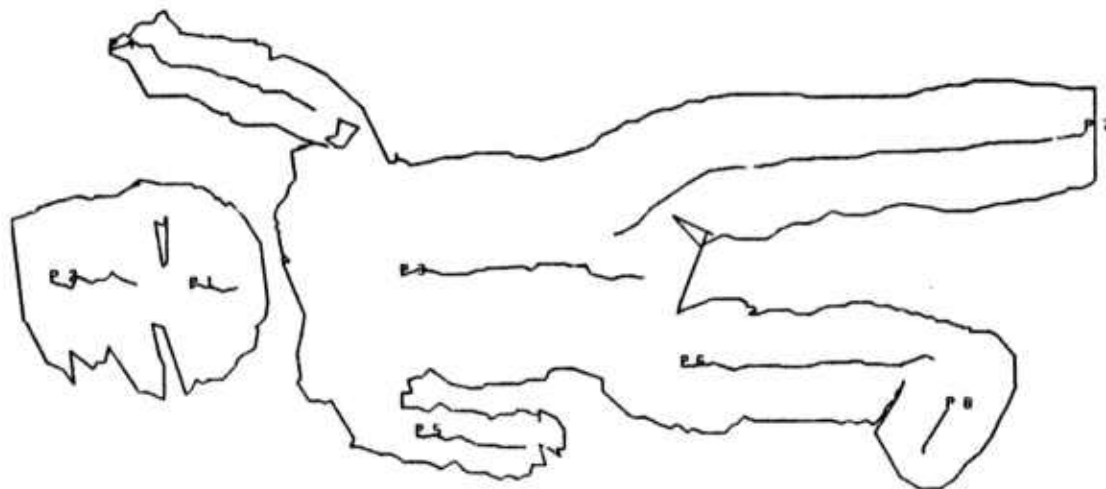


Fig. 7.9 Another View of a Doll

THE MODELS SELECTED BY INDEXING:

HORSE
DOLL

matches in preferred order

DOLL

PRINTING PIECE CORRESPONDENCES

P3 BODY
P7 LEG
P6 LEG
P4 ARM
P5 ARM

NO MATCH FOUND FOR THE FOLLOWING PCS OF THE OBJECT
P8

NO MATCH FOUND FOR THE FOLLOWING PCS OF THE MODEL
HEAD

Fig. 7.10 Recognition Results for the Doll in fig. 7.9

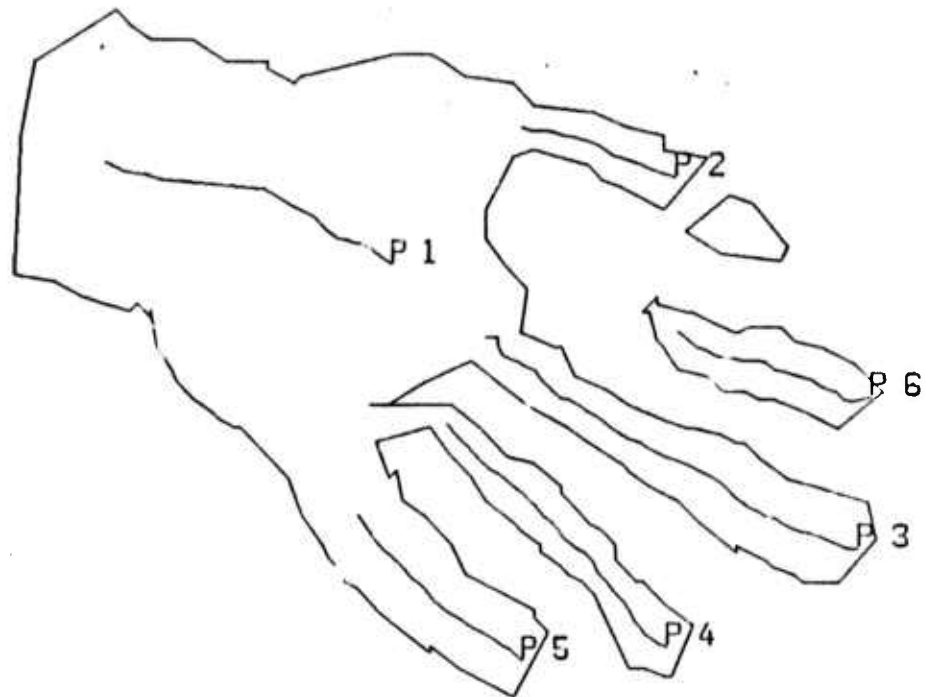


Fig. 7.11 A Glove

THE MODELS SELECTED BY INDEXING:

HORSE
GLOVE

matches in preferred order

GLOVE

PRINTING PIECE CORRESPONDENCES

P1 PALM
P3 MIDDLE_FINGER
P5 LITTLE_FINGER
P4 INDEX_FINGER
P2 FORE_FINGER
P6 THUMB

NO MATCH FOUND FOR THE FOLLOWING PCS OF THE OBJECT

none

NO MATCH FOUND FOR THE FOLLOWING PCS OF THE MODEL

none

Fig. 7.12 Recognition Results for the Glove in Fig. 7.11

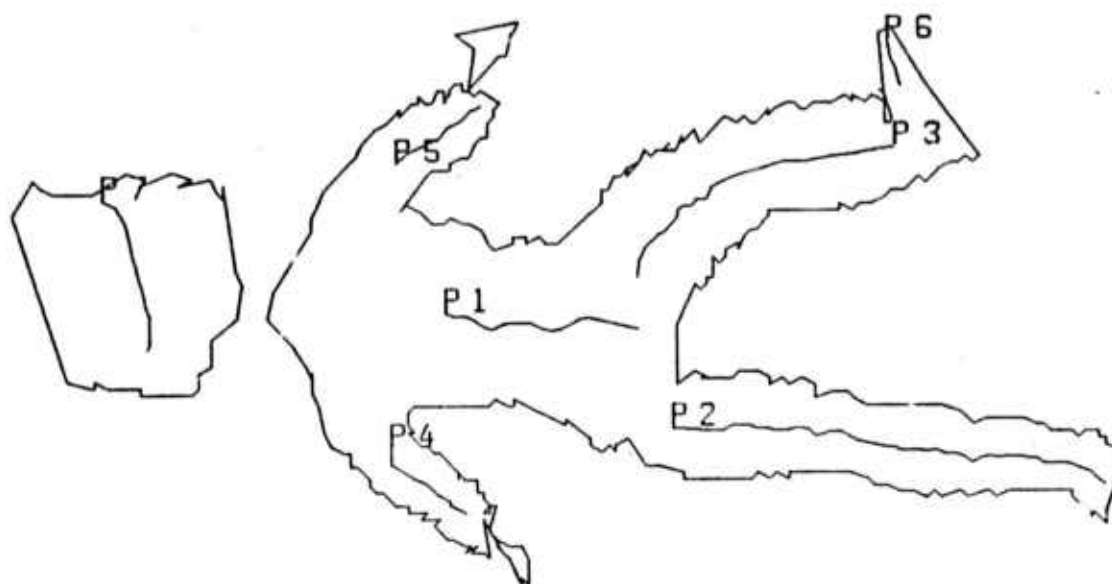


Fig. 7.13 Another View of a Doll

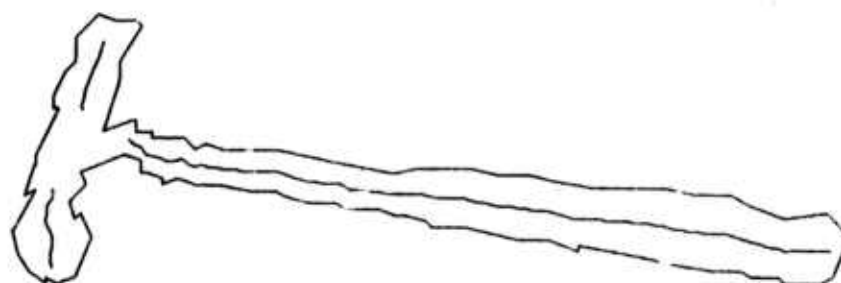


Fig. 7.14 Another Hammer

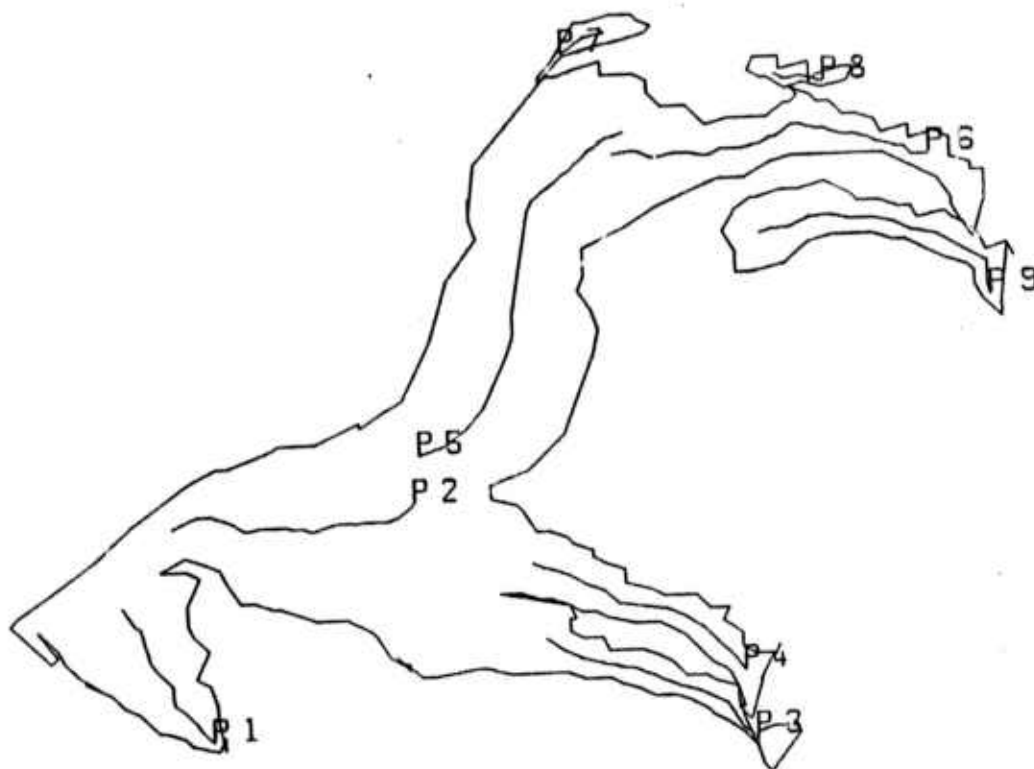


Fig. 7.15 Another View of a Horse

In the cone description process, the local cones are extended until they encounter a discontinuity. The discontinuity is defined locally, as an abrupt change of the cross-section. The local continuity definition works well with cylindrical parts, where the cross-section is constant or varies slowly. Parts segmented by a local discontinuity are merged in later processing, e.g. in Fig. 4.10, pieces P4 and P7 describing the top and the bottom of a leg are merged into a single piece as an alternative description (Also P1 and P2 in Fig. 7.8). The converse, of splitting a piece into sub-parts later can also be useful, but is not implemented. More effort is needed for a better global continuity analysis which takes into account the roughness of the boundaries.

The descriptions of the axes are generally satisfactory in the central parts of a piece. Near the joint of a piece with other pieces, the descriptions can be affected by attempted extensions into parts of the other pieces, leading to either premature termination of a piece description or distortion (curling) of the axis near the end; e.g. see the end of piece P3, in Fig. 7.13, near the joint with the body. This also affects those joint descriptors that rely on the angular relations of the parts. Improvement of such descriptions will require detection of this effect and perhaps redescription after removal of the interfering parts. We are able to detect orthogonal terminations of a piece, e.g. the face of the hammer in Fig. 4.7. Proper description of other terminations will require special routines. More attention also needs to be paid to descriptions near the ends of pieces, e.g. description of a hand terminating an arm. However, more resolution is necessary for the implementation of such descriptions.

The resulting selected descriptions are satisfactory on the whole, the segmentations being consistent with the desired, intuitive descriptions. We believe that the results shown here are for a wide enough variety of scenes, that the success of the programs is *not* attributable to their *tuning* for the specific scenes, and that similar performance can be expected on scenes of similar *complexity*. The description programs have not resulted in any major "extraneous" piece descriptions. The recognition programs ignore small extra pieces (such as the piece P8 representing a foot in Fig. 7.9).

The connections among the pieces are easily inferred from the boundary. Symbolic, summary descriptions are generated for the pieces and the joints of an object, as discussed in sections 5.1 and 5.2. The joint descriptions relying on angular relations of the parts have not been very useful for us, because of the above mentioned uncertainties of the axes directions, and the allowed articulations of the parts. The bilateral symmetry computations rely on very crude measures for part similarities and need improvement.

For occluded scenes, the separation of disjoint bodies is adequate (except for the separation of touching objects). The hypotheses for connections of occluded parts are based primarily on proximity. Implementation of more sophisticated techniques requires improved part descriptions (e.g. more accurate axes directions). Knowledge of support and stability relations can be of value here. No hypotheses are generated for the continuity of a part split into two sub-parts by an occluding part. E.g. in Fig. 4.14, one of the legs of the doll is split into two parts because of the occlusion caused by the snake lying across the leg. Such connection hypotheses will be essential for scenes with heavier occlusion than considered here.

A dark supporting surface (background) has been used for our scenes. The separation of parts of the supporting surface from the objects is not expected to be very difficult when using three-dimensional position data. This problem is included in the problem of separating touching objects, but could benefit from the use of special routines, such as searching for planar surfaces.

The models used for recognition are not ideal, but previously generated descriptions of the objects. No effort has been put in "learning" more complete models. We have used only a small set of models for our recognition experiments. However, the recognition programs are not written for this particular set of models and are expected to work well with other objects, that are well described by the chosen representation. Also, since recognition is strongly dependent on the structure, it is felt that addition of objects with different structures will not adversely affect the performance of these programs (assuming that adequate descriptions are generated). For the following examples, the set of models used consisted of the following: a doll, a horse, a glove, a snake, a ring and a hammer.

The discrimination of the recognition programs is good between objects with different structures. E.g. the glove in Fig. 7.11 is easily recognized by the programs (matches with the glove model generated from descriptions of fig. 4.9). Discrimination of objects from partial structure descriptions depends on the amount of the structure seen. E.g. the recognition of horse in Fig. 7.6 is unambiguous but not in Fig. 7.5, and the doll in Fig. 7.9 is also recognized without any confusion. For objects with similar structures, e.g. a doll and a horse, relative sizes of the parts are used for recognition. With unrestricted limb articulations, the angular relations of the parts have not been useful. For certain viewing angles, the relative size information is not adequate for clearly picking one model over the other (remember, our models are incomplete). In such instances, the multiple choices are reported in their preferred order. E.g. the horse in Fig. 7.3 is recognized even though only a partial view is seen, but the identification of the horse in Fig. 7.5 is not clear (choices of doll and horse are reported, the doll being the marginally preferred choice). Identification of individual parts is an integral part of the recognition process. This makes the problem of further verifying the multiple choices easier (we have not implemented any verification techniques). Note the many articulated views of the same doll, in Figs. 4.12, 6.3 and 7.13; and a view of another doll in fig. 7.9.

In some instances the objects are identified correctly, but the part identifications are in error. E.g. in Fig. 7.6, the tail and the rear leg of the horse have been interchanged (see the output in Fig. 7.7). This is because the decision was based purely on the metric sizes of these parts (the lengths, widths, and the cone angles), and the models had no information about the attachment points of these limbs or the support relations. Shadows can cause part of the structure to be obscured. E.g. in Fig. 4.14, the head of the doll is not seen as connected to the doll body. Without this connection, the recognition programs interchange the identification of the arms and the legs (because of the interpretation of the shoulder piece as head). An hypothesis suggesting connection of the head to the body is generated, but is not examined by the recognition programs because of the above inconsistency. If the head is connected to the body first (by manual intervention), proper identification of the arms and the legs results, with the shoulder piece being classified as an unimportant extraneous piece (since it is not elongated). This example suggests that more "bottom-up" processing of hypothesized connections is likely to be necessary for complex, occluded scenes.

The performance of the recognition programs could be improved by use of more detailed models. Some improvement could also be obtained by more detailed matching of individual parts, instead of just matching the average descriptors. Of course, the use of other data, such as surface color and texture, would simplify many discriminations (e.g. doll *vs* horse). Such data can be obtained from the TV image.

The indexing procedures are successful in working with partial views of an object and

retrieving a list of similar models. The indexing process is "robust" in this sense. Effectiveness of indexing depends on the amount of the object seen, i.e. the number of suggested similar models is smaller when a larger amount of the object is visible. E.g. only horse and doll are suggested as models to be matched against for both the dolls in Fig. 7.5 and 7.9, but the horse in Fig. 7.5 is matched against a doll, a horse, and a hammer. We have used only a few descriptors for indexing; more descriptors need to be added. The number of models used for our experiments is too small to provide meaningful statistical results.

Only generalized cone primitives have been implemented in our description programs. Addition of other primitives, particularly planes, will help in extending their range. We have also not concentrated on the descriptions of objects with holes. The detection of the holes from the boundary information available to us is direct. The modification of the descriptions of the solid part in terms of these holes is more complex.

The speed and memory requirements of our programs are discussed in the following section.

7.2 EXECUTION TIMES AND MEMORY REQUIREMENTS

In the following we present the run times and program sizes for the various stages of processing in our system. All execution times are run times for a PDP-10, KA-10 processor, running under the Stanford Monitor (unpaged) and the programs sizes are for 36 bit words. Estimates of processing times for improved versions of these programs are offered.

The time required for acquiring the laser scan data is essentially determined by the time taken to read a TV frame, and store the non-zero intensity points. We allow 1 second between reading of two frames due to vidicon lag (image persistence). Currently, the time required to scan a scene is about 2-4 minutes. Intrinsically, the time required is limited to that needed for the reading of multiple TV images. With currently available imaging devices, each frame could be processed in 2-3 TV field times (to allow for persistence), and a typical scene requiring less than 200 scans would take less than 10 secs.

Much more time is spent in the preliminary processing stages of the program, than at "higher" levels. This is consistent with the reduction of amount of data at higher levels. Thinning of laser scans takes two to five minutes of runtime, proportional to the number of points in the scans. The program size is about 20K. Computation for thinning is not expected to be reduced by large factors. However, thinning of different laser scans is independent of each other and processing times could be reduced by parallel processing if such processors were available. In our implementation, we use thinning only to locate the end points of segments for linking in a boundary. Unthinned data could be used instead.

The execution time for linking the segments in a boundary depend on the size of the picture. E.g. the horse in Fig. 4.10 required 20 secs whereas the picture in Fig. 3.6 required only 8 secs. The program size is about 20K. The major portion of this processing time is spent in computing the intersections of the two sets of laser cross scans (appendix 1). Such computations would normally be proportional to the product of the number of segments in the two scans. However, the use of laser calibration information limits the number of cross scans that need to be

investigated for intersection with a given scan. The computation time is thus proportional to the product of the number of scans and the average length of a scan. In Appendix 1, we have outlined a method for computing intersections directly by using a large memory array (21.5K) for recording the scans at each point in the image plane. The time required for computation of intersections will be that required to access this array once for each point in the laser scans. For an average scene containing 10,000 points of non-zero brightness, this time is expected to be less than 1 second. Note that this time will be only a linear function of the number of points.

Projection in each direction takes an average of 5 seconds. We project in 8 directions, requiring about 40 seconds. The computation time is proportional to the number of boundary points for the object. The programs are about 30K in size, but the temporary data storage requires up to an additional 50K. Much of this storage could be reduced by more efficient coding of the present programs. The major proportion of the execution time for the projection operations is spent in computing the cross-sections, such as shown in Fig. 4.2. We think this computation can not be reduced significantly, but the projections in different directions are independent and could be computed simultaneously on parallel processors.

The extension of the local cones generated from the projections requires about 45 seconds each for objects in Figs 4.6 and 4.13. The time required to compute these axes is proportional to the total number of cross-sections computed, which is proportional to the length of the axes of the cones. Processing time could be reduced by sampling the axes at coarser intervals. Some parts of the object are described by nearly identical, multiple cones resulting from different local cones, e.g. see the legs in Fig. 4.6. Such duplication could be detected, in some cases, before the extension of the cones by examining the containment of the axes of the local cones. The extension of different cones is independent of each other and thus amenable to parallel processing. The size of these programs is about 30K. Our cone description routines are about an order of magnitude faster than those described by Agin ([Agin 72]), this improvement comes from our use of the boundary rather than the points on the surface in the description process.

The resolution of overlapping cones and the symbolic descriptions of the chosen parts requires less than five seconds for the examples presented here. The matching of a description with one model requires less than 2 seconds. Indexing reduces the number of models to be matched against, in our case to 2 or 3 (the time required for indexing itself is insignificant). With a large model base and no indexing, the matching times would become the major component for recognition. In our implementation, this stage of the processing requires the least time. The symbolic description and recognition programs run in about 60K of memory (the running size of the programs will go up with an increase in the number of models).

The processing speed of the current programs is far from being in "real time"; the average time for complete processing being about 5-10 minutes (including the data acquisition times). However, with the speed up of data acquisition and elimination of thinning this time can be reduced to about 2 minutes. These programs have not been optimized for run time efficiency and improvements can be expected by such optimization, e.g. the elimination of array bound checking and machine coding of the inner loops.

These execution times are, of course, dependent on the speed of the hardware processor. Already, processors five times faster than the processor used for our experiments are available at reasonable costs (e.g. PDP11/45). Processing times of as low as 30 seconds are thus currently feasible. As most of the time is spent in processing that can be done independently and in

parallel, multiple processors can be used to reduce this processing time in direct proportion to the number of such processors. With the expected decrease in the cost of such processors, near real time computation of our algorithms will be feasible at reasonable costs.

The total size of our programs is about 150K, with additional data storage ranging up to 50K, depending on the scene. Much of the processing is sequential and only parts of the programs need reside in the memory at one time. Our system monitor does not permit this and the programs are run as several smaller programs. The size of the programs is not expected to be a major constraint with the use of modern techniques of paging monitors.

7.3 FURTHER RESEARCH

Several improvements in the performance of our programs can be obtained by implementation of techniques suggested previously (in chapters 4, 5 and 6). In summary, the important ones are:

1. Redescription of parts after the descriptions of the neighboring parts are known.
2. Use of more detailed models of the objects, perhaps for specific applications.
3. Verification methods for resolving recognition ambiguities.
4. Incorporation of primitives such as spheres and flat surfaces; and better descriptions of piece terminations
5. Improved resolution from the hardware setup.

In the following are suggestions for further research, related to our work, and to extend the results of this research.

The major need is to extend the results for scenes of higher complexity, such as heavy occlusion or unfavorable viewing angles. Analysis of such scenes is likely to follow a modified control structure. Surface continuity hypotheses will need to be generated at an early stage and communication between different *levels* will need to be more extensive. Analysis would be helped by incorporation of knowledge such as support and stability. Use of such knowledge with only partial information about the objects is unclear.

Simpler analytical techniques may suffice for applications to *visual feedback* where much information is at hand about the expected objects in the scene. Visual feedback has been found to be of great utility in previous attempts at manipulation aimed for industrial automation applications ([Gill], [Bolles]). Incorporation of primitives other than generalized cones is likely to be necessary for objects encountered in industrial applications.

Learning of model descriptions by using several views of the same object and by comparison with other objects is a description learning problem. Winston ([Winston]) approached this problem for the domain of polyhedral objects; we feel that the current domain is richer and presents further important problems. The descriptions generated here are not necessarily perfect

(these problems correspond to missing or extra lines in Winston's case) and some of the relations are metric.

Even though we gather complete three-dimensional data about the visible surfaces of an object, our programs are able to work with merely the boundary data. Such data can, in principle, be obtained from the TV image alone. For situations where the use of laser ranging is not acceptable, the camera image may be the only available input (this still does not preclude the use of depth information obtained by a stereo pair of pictures). We do not expect a 2-D analysis to be easy, but still feel that our techniques offer hopes of making it feasible.

The extraction of boundary information does not require complete three-dimensional position information. *Grid coding* techniques suggest possibilities of extracting boundaries more simply and quickly. Will and Pennington ([Will]) have described experiments with shining various grids on polyhedral objects and direct extractions of plane faces. Consider shining a grid of alternate dark and bright lines on an object. The extremities of the lines on the object can be used to construct the boundaries. However, some ambiguities occur because of coincidence of segments from different lines in the projecting grid (this confusion is what prevents the direct position measurement of all points on the surface by shining a single pattern on the object). Output from shining a rectangular grid is equivalent to that obtained by considering all the laser scans for one scene from our current apparatus at the same time. Fig. 3.3, shows the laser scans for a doll; some scans appear to go unbroken from the head of the doll to the body because of the coincidence of segments from different laser positions. The individual scans, not shown in the figure here, show clear discontinuities and the boundary shown in Fig. 3.4, separates the head and the body. (Note the head and the body are separated because of depth discontinuities from the particular viewing angle.) *Coding* of light patterns on the grid can be used to reduce such ambiguities. For any code, some set of surfaces will give erroneous results. The requirements of a grid code for just extracting the boundary information and not necessarily provide complete range information may be simpler.

7.4 CONCLUSIONS

Research in the area of description and recognition for realistically complicated scenes is preliminary and our programs are not of direct use for applications such as industrial automation. However, we feel that a beginning has been made into extending computer vision techniques to curved and complex objects. We think that our techniques are generalizable, and that for restricted applications at least, extensions of our techniques can be made to work. The performance of techniques presented in this thesis was discussed in detail in sec. 7.1, and suggestions for improvements provided in sec. 7.3. Here we summarize and discuss how our methods relate to some broad issues in computer vision and artificial intelligence.

1. Representation: The power of our programs (or lack of it) is strongly dependent on shape representation. In our experience, the chosen primitives have been useful for the class of objects used in our experiments. We think that they will apply to a broad class of industrial objects and animal shapes. We were able to describe objects at varying levels of detail and differentiate between gross and fine details. The primitives allowed the articulation of limbs to be expressed naturally, and we are able to recognize objects with such articulations. The only intermediate representations of interest were a representation of 2-d image space and a representation of the boundary, particularly proximity on the boundary.

2. Segmentation: The segmentation problem consists of segmenting different objects in a scene and segmenting an object into sub-parts. The body separation problem becomes simpler with the use of depth data but still remains a fundamental problem (as for the cases of touching bodies and supporting surfaces). In this thesis, we have ignored this problem. The notion of continuity is basic to our segmentation of a body into parts. In our case, we have used the discontinuities of a cone description. In the context of the chosen representation, we have been successful in finding gross discontinuities. Better continuity formulations may result in more useful and finer segmentation, and present an important research problem. In our opinion, the low level techniques are necessary, but not all problems need be resolved at that level; the final segmentation decisions can only be made in the context in which they are to be used. The ability to generate alternative descriptions is crucial. We suggested some redescription techniques in the context of a joint.

3. Indexing: Our approach to recognition has been through making descriptions. The descriptive stage seems necessary if the system is to have any indexing capabilities. Some approaches to indexing were presented in this thesis; we believe it to be a major and difficult step in the recognition process, and generally necessary before "high level" knowledge can be used (discussed in more detail later).

4. Matching of Descriptions: Our recognition of objects is by matching two description structures. The description structures contain many descriptors and relations among them. We believe that similarity of such structures can not be adequately evaluated solely by a metric defined on the various descriptors. In our programs, we have chosen to make descriptions of differences and evaluate the differences in the context of the task of recognition (such as, could the two descriptions belong to the same object in spite of the differences by "explaining" the differences). Thus, a three legged horse would be recognized as a horse, assuming the fourth leg to be hidden. Such comparison is essential for recognition from partial information. Also, note that segmented models are essential for such evaluations. The ultimate resolution of the differences must depend on the goals of the program. When confronted with a purple cow, output of a matching process should be that it sees an object with the shape of a cow but the color is in discrepancy. Whether the object should be called a cow or not, must depend on what additional checks can be made and what is the purpose (of making such a decision). For recognition, greater power is available from including model-driven verification tests, e.g. by checking the feet of a horse to distinguish it from a doll.

4. Computational Complexity: The amount of computational effort required is intimately related to the two-dimensional (three-dimensional) nature of the visual processing. Operations such as finding proximity in a plane or space are inherently expensive, but necessary, e.g. for boundary organization, and finding boundary correspondences for the cone descriptions.

5. Implementation effort: An important issue in vision research is the effort required for implementation. Our programs are large and have required considerable investment of time, though they address only a small part of the vision problem (we have dealt with shape only). We believe that at least part of the reason for is in our use of a language like SAIL (or LISP), which operates at too low a level for visual operations. A special "high level" language will aid the system building process.

Use of High Level Knowledge:

Some alternative paradigms, suggesting the use of "high level" knowledge and goal directed ("top down") techniques have been suggested recently. In one form ([Tennenbaum]), this knowledge is used to limit the search, such as by assuming that the telephones are found on tables in an office and by using the knowledge that a table top is easy to find. Another suggestion has been to use hypotheses generated from a very simple description of the scene (or parts of a scene) to guide further descriptions ([Freuder 73a,b]). We think that the principal issue in the use of high level knowledge to guide a vision process is the generation of a workably small number of hypotheses about the scene (or the object). We discuss the various techniques in the context of the following tasks.

1. The visual environment is limited and well known. The properties of objects (such as color) and their approximate locations are known (and no unknown objects are in the environment). Example: a selected office scene.
2. The visual environment is limited and relatively well known. Most of the objects and loose spatial relations among them are known. However, the scene may contain unknown objects.
3. A complex environment with known context, such as an ordinary office or an outdoor scene. The scene contains many complex objects and a complex set of goals is specified for analysis.
4. The context of the environment is completely unknown, such as a randomly selected photograph.

In task 1 above, if the goals and objects are very few (e.g. telephones may be the only black objects on table tops), some relatively simple techniques can be used to recognize objects. For example, Tennenbaum's work uses only pointwise properties such as color and no shape information at all. However, when the environment is more complex (task 2), e.g. black telephones and black note books may be found on a table top, a more detailed analysis (such as shape descriptions) of the scene will be necessary. Tennenbaum's methods can still help in limiting the parts of the scene that need to be described in detail. For known environments, a "top down" approach is likely to be more efficient and less prone to error.

With rich visual environments (such as for tasks 3 and 4), we believe that a sophisticated "low level" analysis will be required to obtain reasonable hypotheses. In our system, the indexing of models (hypotheses) is more effective when a large description structure is available, and local descriptions (of a single cone) can match a very large number of objects. The description mechanisms suggested by Freuder, should be of value after such indexing. We estimate that both low level description techniques and high level recognition techniques will need to be strengthened.

The techniques presented in this thesis are best viewed as modules that can be adapted for specific applications. We believe that such description techniques will be necessary, even in a system based on a top down approach, for anything but simple and known scenes.

APPENDIX I BOUNDARY ORGANIZATION

In this appendix we discuss construction of a boundary from the laser scan data. The laser scans consist of two series of scans. Each scan consists of several TV frames. Each frame corresponds to one position of the illuminating light plane. The frames in one series of scans are produced by light planes which are nearly parallel. The light planes for the two series of scans form an angle of between 60 to 90 degrees (depends on the hardware setup). One series of scans will be referred to as *cross-scans* for the other series.

Points in each TV frame group into a number of *scan segments*. Points within a scan segment are connected, i.e. each point has a neighboring point within one row or one column in the picture. No two points in two separate segments may be so connected. Thus, each scan segment corresponds to a continuous part of the surface. An object boundary is constructed by joining the end-points of scan segments (by straight lines) in an order determined by techniques described below.

An example of the scan output for a doll is shown in fig. 3.3. These scans provide us with the equivalent of a binary intensity digital picture. Construction of the boundaries for such a picture is straight forward ([Duda] pp.290-293). However, in our case, the picture is sampled, i.e. we do not have intensity values at points between the scan segments. The boundary construction needs to estimate whether there is a cavity between segments. Our boundary algorithm is designed to work even if cross-scans are not available. We first describe the algorithm assuming that the cross-scans are available and then describe the modifications.

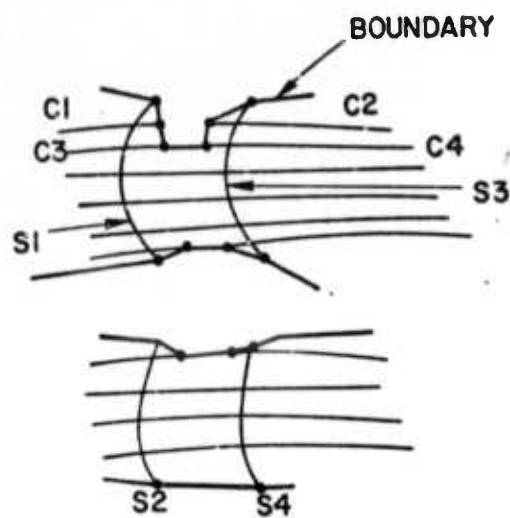
The basic requirement for the boundary is that it not cross a solid part of the object (and hence a scan segment), or a hole. We construct the boundary by comparing segments in successive frames. Two scan segments, S_1 and S_2 , belonging to neighboring frames are considered to *overlap* if there is a cross-scan that intersects both S_1 and S_2 . Overlap of two segments is taken to indicate presence of a solid part of the body between them (since there are some visible points in between). A modified definition of overlap, for situations where sufficient cross-scans are not present, is given later.

We will describe the boundary linking by reference to pictorial examples. We first describe the connections of segments in two successive frames and the cross-scan segments terminating between them. S_i , (i is an integer) is used to denote segments in one scan and C_i to denote segments in the cross-scan. Relations between segments in the two frames fall in one of the following three categories.

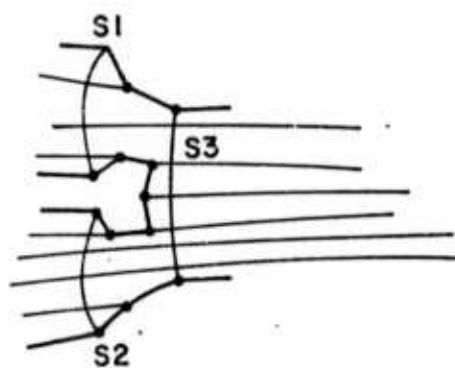
(a) A segment S_1 , in one frame overlaps with only one other segment in the next frame. The linking is as shown in Fig. A1.1(a). Starting from one end of S_1 , we connect the ends of C_i that terminate between S_1 and S_3 until we come to a cross-scan segment that intersects them both. The boundary now connects the cross-scan segments intersecting S_3 but not S_1 , and terminating between S_1 and S_3 , until the end of S_3 is reached.

(b) Two segments, say S_1 and S_2 , both overlap segment S_3 in the next frame. Linking of end-points is as shown in Fig. A1.1 (b).

Note that this generalizes to overlap of many segments with one, by repeatedly considering overlap of two segments in one frame with one segment in the other frame.



(a)



(b)



(c)

Fig. A1.1 Boundary Linking for Different Cases

(c). Segment S1 does not overlap with any segments in the next frame. This situation is shown in Fig. A1.1 (c).

The above cases have illustrated local boundary organization. Starting from one end of the picture and applying the above methods to all frames in one scan will yield one or more closed boundaries. Case (a) above illustrates boundaries of two parts, these may belong to different objects, or join in some other part of the picture. Case (b) illustrates the joining of two such parts. Joining of these parts on the other side also, would indicate a hole. Case (c) illustrates the boundary at the end of a part.

Modifications of the Algorithm.

The algorithm described above is heavily dependent on the determination of the "overlap" of two segments. The requirement of a common intersection cross-scan was used for determination of overlap. However, for short segments we may have no cross-scans to use for determining overlap (because of limited sampling). Also, if the cross-scans are not orthogonal (but at an angle of say 60 degrees), then a single cross-scan may not intersect two nearly parallel short segments (whereas an orthogonal cross-scan would). To overcome these difficulties, overlap is redefined as follows.

Two scan segments S1 and S2 (in successive frames) are taken to overlap if one of the following holds:

1. There is a cross-scan segment that intersects both S1 and S2.

or

2. (a) There is no cross-scan frame, such that one segment of this frame intersects S1 and the other intersects S2, and

(b) There exists a plane through the point of illumination and normal to the plane of S1 that it intersects both segments S1 and S2.

Example: Fig. A1.2,

(a) S1 and S2 overlap

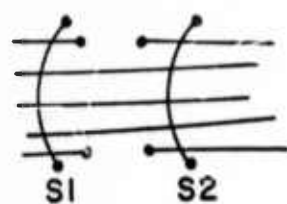
(b) S1 and S2 do not overlap (C1 and C2 belong to a common frame; C3 and C4 belong to another common frame).

(c) S1 and S2 do overlap (C1 and C2 do not belong to the same frame). between them.

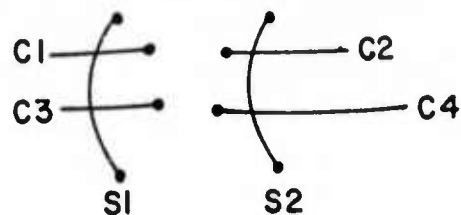
(d) S1 and S2 do not overlap (condition 2(b) is not satisfied).

If condition 2(a) is not satisfied, it indicates there is evidence of a break between S1 and S2. Condition 2(b) indicates that an orthogonal cross-scan could have intersected them both and with lack of evidence to the contrary (i.e. condition 2(a)), we assume that S1 and S2 do overlap.

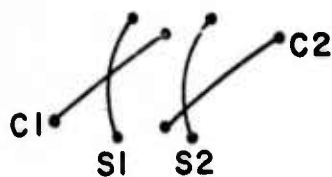
This modified definition is equivalent to assuming that there is no boundary between



(a)



(b)



(c)



(d)

Fig. A1.2 Examples of Segment Overlaps and Non-overlaps

two scan segments when we have no way of testing it with the available data. Improvements could be obtained by using the gray level TV image in addition to the scan information; we have not implemented this. The resulting boundary is affected by any errors in the input data, such as missing scan points or additional scan points caused by noise. Some degree of resistance to such errors is achieved by ignoring boundaries formed solely by connecting segments in just one frame (this overcomes holes caused by a few missing points in just one segment or boundaries outlining isolated noise points). Note that these difficulties of boundary detection from laser scans are minimal compared to the difficulties of boundary detection from gray level pictures. Further, these problems are resolvable by improved scanning hardware.

Another problem, caused by some unanalyzed attributes of our scanning apparatus, has been in the end-points of the two series of scans not corresponding exactly, but offset from each other. This makes the resulting boundary jagged and affects the accuracy with which cone descriptions can be generated based on these boundaries.

Computational Requirements.

In our method, boundary linking requires calculating the intersections of the segments from the two cross scans. We need calculate only a few of the possible intersections near the ends of the segments. Given a segment S_i , we can calculate which cross scan angles can contain segments that intersect S_i (by calculating the angles of end points of S_i from the laser viewpoint, and obtaining cross scans in that range of angles). The intersection of two segments is determined by making piecewise linear approximations to the two segments. A few minor errors and extra effort are caused by slight errors in the intersection process and by the calibration uncertainties which give small angle errors in choosing cross scans.

An alternative approach for computing intersections would be to use a large array, where each byte of the array corresponds to a position in the image plane. For each point that belongs to some segment in one scan orientation, mark the corresponding byte in the array by this segment number. Now, for each point that belongs to some segment C_j in the cross scan, check whether the corresponding byte in the memory is marked. If so, then the segment C_j intersects with the marked segment in the memory, at this point. This method will give us all intersections without searching. The obvious disadvantage is the requirement of a large memory. (Each point in image plane requires only enough bits to identify the segments in one scan. Nine bits per point would be adequate for us. In our system, with 333×256 image points, a storage of 21.5K, 36 bit words is required, which is not prohibitive. The size can be further reduced by variations of hash coding.) The time requirements of this method will be proportional to the boundary length.

The computational effort required for computing intersections is inherent in the two dimensional nature of the problem. versus time tradeoff between the two methods described here. Actual computation times for boundary construction and expected improvements are discussed in sec. 7.2.

APPENDIX 2 DESCRIPTION OF ALGORITHMS

This appendix presents the algorithms used in this thesis. These algorithms, the reasons behind choosing them and their performance have been discussed in the previous chapters. The purpose here is to state them in one place in a concise form with all relevant details. The appendix is divided in the following sections paralleling chapters 4,5 and 6.

1. Segmentation: Projection, Extension of local cones, Descriptors for a cone, and selection of segmentations.
2. Symbolic Descriptions: of pieces, joints and object. Shadowed pieces.
3. Recognition: Indexing, matching and choice of matches.

A2.1 SEGMENTATION

SOME DEFINITIONS:

Definitions of some terms used in the following algorithms are repeated here. The term "cone" will mean a "generalized cone". A three-dimensional (two-dimensional) generalized cone is generated by sweeping a planar (linear) *cross-section* along a curved line in 3-space (2-space). For a two-dimensional cone, the term "cross-section" will mean a straight line segment entirely within an object, terminated by *end-points* on the boundary of the object. For three-dimensional cones, the *end-points* of a cross-section will mean the two point of the cross-section that are on the boundary of the visible surface of the cone.

A2.1A. PROJECTIONS.

The projection procedure finds two dimensional local cones in the camera image, using the boundary. The following projection procedure is repeated for 8 directions, X_1, X_2, \dots, X_8 , at 22.5 degrees interval from 0 to 180 degrees. The following describes the projection procedure for one of these directions, say X_i .

1. Transform Co-ordinates:

Transform the coordinates of the points on the boundaries of the object to a system with axes X_i, Y_i ; where Y_i is orthogonal to X_i . (Note, the figures to be presented here are all drawn with X_i pointing horizontally.)

2. Form Cross-sections:

Form "cross-sections" (two-dimensional), parallel to Y_i , such as shown in Fig. A2.1 by solid straight lines, at regularly spaced intervals of 10 picture units (the complete picture is 330 units wide). Fig. 4.2 shows cross-sections for an actual example (Note that all cross-sections are not exactly parallel to the Y_i axis, because the ends of cross-sections are limited to points on the boundary obtained from actual laser scan data.) The cross-section end points are stored in an array of lists indexed by the X_i coordinate.

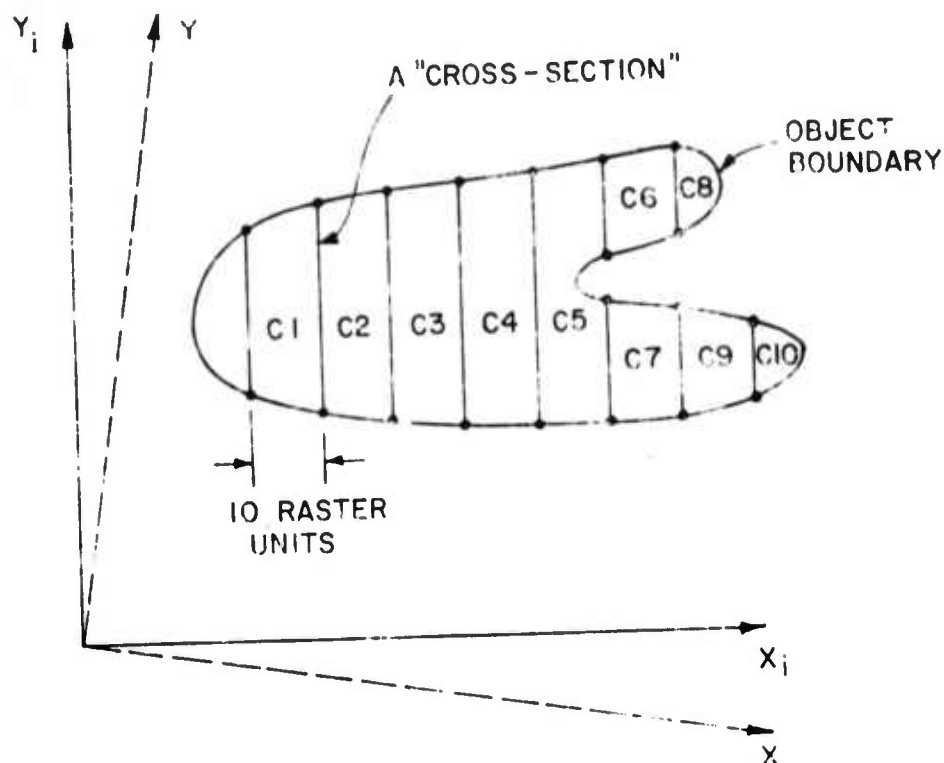


Fig. A2.1 Cross-sections for a Selected Projection Direction

3. Test cross-sections for local cones:

(a) Find neighboring cross-sections:

Two cross-sections are neighboring if *each* end of one is connected to one of the ends of the other through any part of the boundary and no other cross-section ends lie along this part of the boundary. For example, in Fig. A2.1, C1 and C2 are neighbors and C7 and C9 are neighbors, but not C5 and C7, nor C6 and C7.

(b) Test for local cones:

If the line joining the mid-points of a pair of neighboring cross-sections forms an angle of less than 22.5 degrees with X_i , then these two cross-sections are taken to form a local cone.

If either of the cross-sections belongs to a previous local cone, add the other cross-section to the same cone, otherwise include these two in a new cone.

(c) Repeat steps (a) and (b) for all pairs of neighboring cross-sections.

4. Retransform the co-ordinates (of the mid-points of those cross-sections that are part of a local cone) to the original (X,Y) coordinate system.

The output of this procedure is a set of local cones. Each local cone is defined by: a list of the mid-points of the cross-sections comprising this cone and the parts of the boundary on the two sides of the cone. Fig. 4.3 shows the axes of the local cones obtained from the cross-sections of fig. 4.2. Fig. 4.4 shows the axes of local cones of the same doll, obtained by projection in 8 different directions. Note that various parts of the body are described by different local cones for different projection directions.

A2.1B. EXTENSION OF LOCAL CONES:

The projections provide us with a number of local cones. In this section we describe procedures to extend these cones over larger parts of the object continuously. Before extension can proceed, however, we "refine" the axis of the local cones as explained below.

Axis "Refinement":

The axis of the local cones generated by the projection procedure is not necessarily normal to the cross-sections (is within 22.5 degrees of it). Example, Fig. A2.2, shows cross-sections C_1 and C_2 constructed during a projection, and the local cone axis given by joining their mid-points, M_1 and M_2 . The line M_1M_2 was required to be only within 22.5 degrees of being normal to C_1 and C_2 . Starting from here, we wish to find another axis, and cross-sections such that:

1. The cross-sections are normal to the axis.
- 2) The axis passes through the mid-points of the cross-sections.

First we illustrate the procedure, by using the above example. Starting from the axis M_1M_2 , we compute new cross-sections, C_1' and C_2' , normal to M_1M_2 as shown in Fig. A2.2. LET M_1' and M_2' be the mid-points of these new cross-sections (not shown in the figure). Joining M_1' and M_2' , we get a new axis and this process is repeated until no significant changes occur. (Note that in our example, the process converges immediately after one iteration, as M_1' and M_2' are indistinguishable from M_1 and M_2).

In the above example, we have described the procedure in 2-d, for the sake of clarity. Actual computations are performed in 3-d space. The cross-sections are now a planar area, however, we will compute only the end-points of the cross-sections (i.e. the points of the cross-sections on the boundary). The following describes more details of the algorithm used.

Details of axis refinement:

This algorithm starts from the local cone axes given by the projections. The axis is specified as a list of points, known as *axis points*. The 3-d positions of the points are used (obtained from the 2-d positions by a known calibration). Only the part of the boundary that is

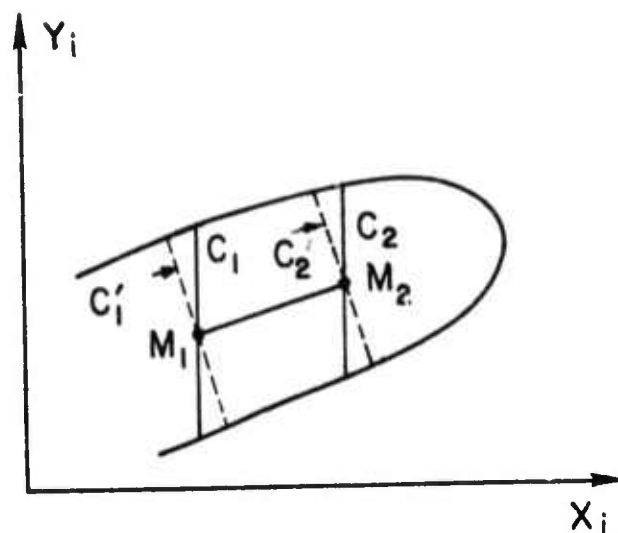


Fig. A2.2 Axis Refinement

associated with the local cone, as supplied by the projection procedure, is used here. The goal of this process is to compute an axis and a set of cross-sections such that:

- 1). the axis is normal to the cross-sections and
- 2) the axis passes through the mid-points of the line joining the end points of the cross-sections.

The following are the steps in this process.

1. Compute axis direction: Fit a straight line to the axis points (Least mean squares fit).
2. Construct New Cross-Sections: At each of the axis points, construct a plane normal to the axis (as determined in step 1 above). Compute intersections of each plane with the given parts of the boundary on the two sides. These intersections for each plane constitute the end-points of a new cross-section. Note that a normal plane at some axis point may not intersect with the given parts of the boundary on either one or both sides. In either of these cases this normal plane does not contribute to a new cross-section.

If the number of new cross-sections falls to 1 or less (by not being able to find boundary intersections) then this procedure terminates and no cone results from this processing (the local cone we started from is rejected for any further processing).
3. Compute new axis points: Compute the mid-points of the end-points of the new cross-sections constructed in step 2. These mid-points form the new axis points. For each new axis point, compute its distance from the old axis point. Compute the average of these distances, call it the *average correction*.

4. Evaluate the new axis and cross-sections: One of the following steps is taken next.

- a). Acceptable cone: If the average correction computed in step 3 is less than the 3-d distance corresponding to 2 picture units (is related to the expected ranging error), then the axis refinement procedure terminates, with its output being the new cross-sections and the new axis points.
- b). Iterate: If the average correction is larger than accepted in (a), then iterate by going to step 1 (unless (c) below holds). Use the newly found axis points (in step 2) for iteration.
- c). Accept the cone: If the number of iterations caused by (b) equals five, the procedure is terminated and the new axis points and the new cross-sections are accepted (regardless of the computed average correction).

The output of this algorithm is: a list of the axis points; a list of the end points of the corresponding cross-sections; and the associated boundaries of the cone.

Extension:

The cones from the above process are extended continuously at both ends. We will first illustrate the extension by an example, and again use a two-dimensional example for clarity. Details of the algorithm follow shortly. Fig. 4.5, shows the axis of a cone to be extended. The axis is extrapolated a small distance and a normal cross-section constructed. The mid-point of the new cross-section and its distance from the extrapolated axis point is computed. If this distance is small the axis extends to this mid-point. If the distance is large (as is the case in the example), a new direction for the extrapolated axis is computed by including the new mid-point. A new cross-section normal to this new direction is constructed. This process could be iterated but we use only one iteration. The new cross-section is then examined for width continuity.

The details of the extension are described in the following. Extension at one end is described here, and the procedure is repeated for the other end.

1. Extrapolate the axis: Starting from one end extrapolate the axis in the local direction (computed by fitting a least mean squares straight line through the last five points) by an amount equal to 0.025 the width of the cross-section at this end (limited by a minimum of the 3-d distance corresponding to 2 picture units and a maximum of the distance corresponding to 8 picture units).

2. Construct a new cross-section:

Construct a plane normal to the axis at the extrapolated point.

- (a) Is the end of object reached?

Examine whether any part of the boundary of the object is beyond this plane in the direction we are extending, if not then the end of the object has been reached and we try a half step (step 6).

- (b) Compute intersections of the plane with the boundary on both sides of the cone. The

intersections determine a new cross-section and the mid-point determines a new axis point. If no intersections can be found with the part of the boundary already associated with the cone, then an additional length of boundary (5 more boundary points) is added. If addition of three such boundary lengths still do not produce an intersection then we try a half step (step 6). (Such condition indicates the boundary has changed direction abruptly, e.g. no intersections with the boundary may result when extensions of a cone describing an arm of the doll of Fig. 3.2, into the body are attempted, because of the sharp turn of the boundary. Note that further tests apply, even if the intersections do result.)

3. **Examine the new Axis point:** If the distance of the new axis point from the extrapolated axis points is within a distance corresponding to 4 picture units then proceed to step 4. Otherwise, fit a straight line (least mean squares) to this new axis point and four previous axes points to define a new axis direction. Go back to step 2 (to get a new normal cross-section). After one such iteration, accept the new axis point regardless of the distance. Note that this step allows the cone axis direction to change.
4. **Examine boundary Crossings:** Check that the straight line segment joining the two end-points of the new cross-section does not intersect with the boundaries at other than the two ends. (This prevents the extension of a cone into holes.) If so, then go to half step (step 6).
5. **Examine the Cross-section width** (defined to be the distance between the two end-points): The width of the new cross-section is predicted by a least mean squares, parabolic fit to the previous five cross-section widths. If the predicted width differs from the width of the current cross-section by less than $0.25 \times$ current width, then this cross-section is acceptable and extension continues (step 1). Otherwise try a half step (step 6).
6. **Half Step:** This step is reached because extension with the extrapolated cross-sections failed at either steps 2,3,4 or 5. Step 1 is repeated once with half the normally used distance for extrapolation. If the current step was already a half step, extension terminates (for one end of the cone). The use of this half step allows sharp changes in the cone to be sampled closer and also to define the terminal cross-sections more accurately.

Each local cone is extended in both directions. The output of these procedures is represented as follows: a list of the position of the axis points; the direction of the axis at each of these points; the widths of the corresponding cross-sections; and the boundary segments on the two sides of the cones.

Fig. 4.6 shows the axes of the extended cones for all the local cones of Fig. 4.4. Note that several areas of the object are described by more than one cone. This is clear for the cones describing the head. However, extensions of different local cones for the same leg or the same arm have resulted in nearly identical cones. In Fig. 4.6, the axes of all extended cones are shown together and the different overlapping cones for the legs and the arms are barely distinguishable. Elongated parts have cone descriptions with well defined axes and is the reason behind various local cones converging to similar cones when extended. The resolution of these multiple descriptions is discussed next.

A2.1C. DESCRIPTORS FOR A CONE:

The following summary information is stored for each cone.

1. Length of its axis.
2. The average width of the cross-sections (the width of a cross-section is defined to be the distance between its two end-points).
3. The average cross-section widths at both ends of the cone (computed by averaging the widths of the last five cross-sections at each end).
4. Length to width ratio (will be called LWRAT)-Ratio of the quantities defined in 1 and 2 above.
5. Average Cone Angle - The cross-section width at a point is a function of the length of the axis up to that point. This function is approximated by a straight line. The slope of this line determines the average cone angle.
6. Axis direction and position at each end of the cone (the cross-section at each end will be known as a *terminal* cross-section).
7. The associated boundary with the two sides of the cone (stored as lists of boundary points).

A2.1D. CHOICE OF SEGMENTATIONS:

Any two cones which share any common part of the boundary are compared to find the extent of their overlap. If the two cones describe much the same part of the object then one of the cones is selected based on chosen preference criteria and the other cone is eliminated from further consideration as the choice for representing a part of the object.

The following tests are performed on the two cones in the order presented below (an example follows this description):

1. Is one of the cones too short?

If the LWRAT for either cone is less than 0.33, the cone with the smaller LWRAT is eliminated (short cones are assumed to be of not much interest if a longer cone describes part of the same area).

2. Is one cone largely included in the other?

Compute whether more than 0.75 of the boundary points on each side of one cone are included in either boundary lists for the other cone. Note it is possible for both cones to be so included in each other. Also make lists of boundary points for each cone that are not included in the other.

If only one cone is so included, eliminate the included one.

If both are included, one of them is selected as explained in step 4.

3. Is one cone included in an extension of the other?

Consider evaluation of whether cone1 is included in the extension of cone2. Let the "center" of a cross-section be defined to be the mid-point of the straight line joining the end-points of the cross-section. Construct two straight, circular cylinders, with their axis starting from the centers of the two terminal cross-sections of cone2 and pointing away from cone2. Let the length of the axis of the new cylinders be $0.35 \times$ length of the axis of cone2 and the diameter be the width of the terminal cross-section at that end plus the 3-d distance corresponding to 10 picture units. (These cylinders are meant to be approximations of how cone2 would extend at each end with the radius enlarged to accommodate the effects of various errors).

If each boundary point of cone1 that is not included in cone2, lies within the volume of either of the cylinders described above, then cone1 is taken to be included in the extension of cone2. Again, both cone1 and cone2 can include each other this way (see step 4). If only one cone is included, the included cone is eliminated.

4. If both cones include each other, then one of the cones is picked based on its descriptors. The basis for choice are the length to width ratios of the two cones or their average cone angles, depending on which gives better discrimination. Compute the ratio of the larger cone angle to the smaller cone angle and call it CONE RATIO. Compute the ratio of larger LWRAT to the smaller LWRAT and call it the LW RATIO. If the CONE RATIO is higher than LW RATIO then pick the cone with the smaller cone angle else pick the cone with the larger LWRAT.

Example: Fig. 4.6 shows the extended cones for a doll and Fig. 4.12 the selected cones obtained by the above procedure. Most overlapping cones were resolved by step 2 in the above (i.e. the selected cones included the other cones describing the same area), with the following exceptions. The cone representing the shoulder area in Fig. 4.6 was judged to be included in the extension of the cone representing the body area (P1 in Fig. 4.12). Note that one of the legs in Fig. 4.12 consists of two cones, P5 and P6. Here, P5 was not judged to be included in the extension of P6 (extended P6 does not include the lower part of P5). However, P5 and P6 are judged to be continuous and merged into a single piece later (this is described in "Merging of two Pieces" in sec. A2.2B of this Appendix).

We have used the inclusion of boundary in step 2 above. Use of area inclusion was discussed and recommended for future use in sec. 4.5. Also, the evaluation of the cones by extension in step 3 is not very robust for resolving cones like the shoulder cone in the doll example. Redescription techniques expected to give more reliable results were discussed in sec. 4.5 (but have not been implemented).

Note that in the above processing of two cones with overlapping boundaries, one of the cones is not necessarily eliminated. The resulting segmentation having parts with small overlaps is quite acceptable.

A2.2. SYMBOLIC DESCRIPTIONS:

Some Definitions:

A *piece* is defined to be a segmented sub-part of an object. Pieces are represented by cones in the current implementation. However, the term will be used to allow for addition of other types of primitives.

Different pieces are connected at *joints*. (Joints are discussed in more detail below).

A2.2A. PIECE DESCRIPTIONS

Descriptions used for a piece were covered under the heading of cone descriptors, sec. A2.1C of this appendix.

We now discuss the construction of the joints; the descriptions of the joints; and the descriptions of a complete object. In the following, we provide the details of only those descriptors that we have found useful in our recognition programs. Details of other useful descriptors for future use are covered in chapter 5.

A2.2B. JOINT DESCRIPTIONS

The joints of pieces (represented by cones) are determined from the connections of their terminal cross-sections along the boundary. The joint construction is discussed only by pictorial examples. Fig. 4.12 shows the selected cones for a doll. Fig. 5.2 shows the joints constructed for these cones (the joint area is shown shaded). Joint J1, between the cones P1, P4 and P6 is an example of the joint where the cones at a joint do not overlap each other. The boundary of this joint includes part of the boundaries of the object and the terminal cross-sections of the cones. Joint J2 is similar; note that P7, representing the head is not connected to this joint (it belongs to a separate boundary because of shadows). Joint J3, between P5 and P6 shows a joint between two intersecting cones. Joint J4 shows the joint of a single cone (this cone did not extend to the end of the leg and the uncovered area constitutes the joint area; cones for the arms and the other leg do extend to the end of the object and hence are not attached to any joint at one end).

Two pieces (cones) belong to the same joint if one of the following holds.

- (a) The part of the boundary between an end-point of the terminal cross-section of one piece and an end-point of the terminal cross-section of the other piece is not included in any other piece.
- (b) The two pieces include a common part of the boundary.

The following descriptions are associated with each joint:

1. A list of the pieces attached to this joint and their order (the order is determined by the order of the pieces as they appear in the image plane). Example: Order of pieces at joint J1 in Fig. 5.2 is P1, P4, P6 or the reverse. We do not differentiate between the two orders.
2. The widest piece of the joint.

3. The relative widths of the pieces connected to this joint (normalized by the width of the widest piece).
4. Joint types: the joint types were described in sec. 5.2. No further details are provided here, as we have not used them for recognition, also for the reasons stated there.

Merging of Two Pieces:

If a joint has only two attached pieces, the following tests are made.

- (a) Are the terminal cross-sections of the pieces at the joint similar (measured by their widths being within 25 percent of each other) and
- (b) Is the joint boundary covered in the extensions of both pieces (in the same sense as in step 3, choice of segmentations) ?

If both of the above are true, then an alternative description is made by merging the two pieces. The cone descriptors are recomputed for the merged cone. Note, the recognition programs investigate both alternatives and choose the one providing the better match.

A2.2C. OBJECT DESCRIPTIONS:

Some descriptions for the complete object are made. Only those descriptions that have been used for recognition will be described here.

1. Number of *well defined* pieces (a well defined piece either has a length to width ratio of > 3.0 or is a distinguished piece, as described below).
2. Distinguished piece descriptions: Distinguished pieces are determined in the following way:

(a) Determine the two widest pieces of the object. If the ratio of the widest piece to the next widest is larger than 2, then the largest piece is a distinguished piece. Otherwise both are considered distinguished. Such distinguished pieces are defined to be of type Wide. Example: for the doll, both the head and the body are such pieces.

(b) If one piece has length to width ratio larger than twice the length to width ratio of every other piece of the object, then this piece is defined to be distinguished and of type Long. Example: for a hammer, the handle qualifies as a long distinguished piece.

If one end of the distinguished piece is different from the other end, in one of the following ways, this property is also associated with the distinguished piece (helps in matching).

- (i) The width of the widest piece connected at one end is more than twice the width of the widest piece at the other end.
- (ii) The maximum of the length to width ratio of the pieces connected at one end is more than twice the maximum of the length to width ratio of the pieces connected at the other end.

3. Bilateral Symmetry: See section 5.3. Not discussed here because not used for further processing.

A2.2D. SHADOWED PIECES:

Connections of shadowed pieces to the rest of the object are not directly available from the boundary data. A piece which is unconnected at both ends, will be called an isolated piece. We determine, which ends of an isolated piece are in shadow of some other parts of the object (by converting the positions of the points on the object to a system centered at the laser source, and determining whether part occludes the other). Connections for each isolated piece are investigated as follows.

For each shadowed end, compute the closest distance of the terminal cross-section from the boundary of each joint. The shadowed end that is closest to a joint is picked and is hypothesized to connect to this joint. The recognition programs investigate the connection of the shadowed piece according to this hypothesis. If a satisfactory match is found for this piece with this assumed connection during recognition, then the connection hypothesis is considered to be confirmed. If this fails, no further attempts are made to determine the connectivity of this piece. More sophisticated alternatives for hypothesizing connections and difficulties in their implementation are discussed in sec. 5.4.

A2.2E. NOTES ON DATA STRUCTURES.

The connection of different joints and pieces is stored. For each joint we store the other joints that it is connected to, and the piece connecting the two. Also stored are the piece, joint and object properties as described previously in this section. Note that we have complete information about the structure, and summary descriptions of pieces and joints. The angles between different pieces at a joint are not explicitly stored but are computable from the axis directions at the ends of pieces. The angle information has not been used in our recognition programs, as we have allowed free limb articulation. The angle information is of obvious value in the recognition of unarticulated objects, or where the articulation limits are known. Computer storage of descriptions is discussed in detail in Appendix 3.

A2.3. RECOGNITION:

In the following we present descriptions of algorithms used for recognition. A detailed example is in sec. 6.1, and performance results are discussed in chapter 7.

A2.3A. INDEXING:

The first step in recognition is to find a set of likely models. The following describes the implemented algorithm. This indexing scheme has been a preliminary effort and extensions of the method are discussed in sec. 6.7.

For each distinguished piece of an object or a model, we form a "description code" consisting of the following three bits.

1. Is the distinguished piece connected at both ends. A model piece is connected at an end if it is connected to any other piece at that end. An object piece is so connected, only if it is connected to a *well defined* piece at that end (for reasons discussed below).

2. Is this piece conical (is the average cone angle > 0.3 radians).
3. Is the type of distinguished piece long (otherwise it is of type wide)

Let $N1$ be the number of pieces connected to one end of the distinguished piece and $N2$ the number of pieces connected at the other end (again use *well defined* pieces if considering the object to be recognized). Let $N2 \leq N1$. All models having the same description code are stored in a list ordered by the value of $N1$.

For each distinguished piece of the object, retrieve the list of models having the same description code. Search along this list for models whose values for $N1$ and $N2$ are at least as large as for the object. Since the list is ordered, we need search along this list only until we find the first model with value of $N1$ smaller than acceptable.

Here, we are assuming that the object can not contain more pieces than the model, but can have an arbitrary number missing because of occlusion. Also only well defined pieces have been used in counting the pieces for the object, since such pieces are elongated and unlikely to appear erroneously in the descriptions.

If the object distinguished piece is connected at one end only, we also retrieve the models that would have been obtained, assuming that it were connected at both ends. Further modifications to the description code could be made based on the confidence with which the distinguished piece is known to be conical (this is not implemented). As a further test, if the total number of well defined pieces of a retrieved model is less than the total number of visible, well defined pieces of the object, then this model is not considered for a match.

For an estimate of the effectiveness of the current implementation, and its extensions, see sec. 6.7.

A2.3B. MATCHING:

The object description is compared to each model found by indexing. A *match* is defined to mean a set of correspondences between the pieces and the joints of an object description and the pieces and the joints of a model description. Associated with each match is a description of the differences. The matching process is described below, the first four steps are concerned with matching the object description to one model description.

1. Form a pair of distinguished pieces, one each from the model and the object description: One match results from each such pair. Steps 1 through 4 are carried out for each pair of distinguished pieces that satisfy the following conditions.

- (a) The type of the two distinguished pieces (such as long or wide) is the same.
- (b) If the object piece is connected at both ends, then so is the model piece.

Compute the *piece match error* resulting from matching this pair and associate with the match. Piece match error is an evaluation of how well two pieces match, and is a function of the differences in the widths, lengths and the average cone angles of the two pieces; this function is fully specified in sec. 6.3.

NOTE: The average width of each distinguished piece used to form a pair here is used to normalize the sizes (lengths and widths) of the other pieces belonging to the same object or the same model. This makes the following matching process insensitive to scaling of the complete object.

2. Match the ends of the given pair of distinguished pieces: Let the distinguished pieces be called MDP and ODP. Let the joints at the two ends of MDP be MJ1 and MJ2; and at the ends of ODP be OJ1 and OJ2. These joints can be matched in two ways:

- (i) MJ1 with OJ1, and MJ2 with OJ2 or
- (ii) MJ1 with OJ2, and MJ2 with OJ1.

We test whether one or the other can be selected without a complete evaluation of these matches (as in step 3 below). The following tests are performed to determine whether the two ends of ODP and MDP are "unsymmetrical" in the same way:

- (a) If both ODP and MDP have the property that the average width at one end is at least twice the average width at the other end, then the match with corresponding ends is selected.
- (b) If both pieces have the property that the widest piece attached to the joint at one end is at least twice as wide as the widest piece attached to the joint at the other end, then the corresponding ends are matched.
- (c) Similar to (b), but use the length to width ratio of the attached pieces, instead of the width.

If a choice between (i) and (ii) above can not be made here, then in the following step, we evaluate both alternatives, else we evaluate only the chosen alternative.

3. Evaluate matching of distinguished pieces, with specified joint correspondences:

This involves matching the two pairs of joints (such as MJ1 with OJ1 and MJ2 with OJ2). Now consider matching of two joints (such as MJ1 and OJ1). With each joint is associated an ordered list of pieces attached to it. Note that the distinguished pieces attached to these joints (MDP and ODP) have already been matched. The remaining pieces are matched with the following constraints:

- (i) the order of pieces in one list must be the same (or reverse) as the order of the pieces they are matched to in the list.
- (ii) the matching is "optimal".

The optimal matching was described in sec. 6.3 and is not repeated here. It is based on picking best matches for individual pieces and minimizing the total piece match error.

With each matching of two joints, we associate the resulting piece correspondences and the following evaluations of the match:

(i) the average and maximum piece match error of the pieces matched.

(ii) a list of the unmatched pieces of the object and the model that are connected to the matched joints.

If a shadowed object piece is hypothesized to connect to the object joint being matched (as explained in sec. A2.2D), then match this piece to an unmatched piece of the model joint (if any). If the piece match error of this match does not exceed the maximum of the piece match errors of the other pieces matched at this joint, then associate this match with the joint match. Otherwise, ignore this match.

A pair of joint matches is simply represented by two joint matches as described above.

NOTE: the evaluation of these matches does not contain a term depending on the angles between the pieces, i.e. the articulation of different pieces is assumed to be completely free. Also note that the length and width of the pieces are relatively insensitive to articulation, since we are using 3-d data. Small variations may occur in the width depending on what part of the cross-section is seen.

4. Select one pair of joint matches: If in step 3 above, we evaluated both pairs (suggested in step 2) then one of them is selected now (based on the above described evaluations, details are provided in the next section of this appendix). If one pair can be selected confidently, then the other one is not used for further matching. Otherwise, we maintain an ordered list containing both options.
5. Choose between different matches resulting from different pairings of distinguished pieces (the selection procedure is the same as used in step 4). If some pairings are clearly preferable, only these are retained. All retained pairings are arranged in a list ordered by their preferences. Note that the models that are very different from the object are excluded from further consideration and this list is not expected to be long even if many models are present (the indexing procedure is expected to have already reduced this number considerably).
6. Steps 1 thru 5 are repeated for each model with which the object is to be compared. The preferred models are selected based on best match with each model (again, the same selection procedure applies).
7. Extend matches: In step 6, we were choosing based only on the matching of a distinguished piece, its two joints and associated pieces. Now each match is extended to include other pieces if any.

For each pair of pieces matched so far, at least the joints at their one end were matched. If the joints at the other end are not matched, do so now. This may require the matching of a joint against a null joint, if one of the pieces is connected to other pieces at one end only. This process is repeated until all joints (and hence all pieces) have been matched.

8. Choose between matches with different models again, based on the extended matches. Output the preferred matches in order. One or more models may result as the output of this recognition scheme.

The selection between two matches is discussed next.

A2.3C SELECTION OF MATCHES:

A match between two descriptions consists of a number of joint matches. The joint matches were described in the previous section. The selection of matches is independent of the number of joint matches included, and hence the same procedure is applicable for choosing at different stages of matching.

The following quantities are computed for a match from its component joint matches:

- (i) total number of *well defined* pieces of the object that are not matched (called NEXCESS).
- (ii) The average piece match error (averaged over all joint matches), called AVERR.
- (iii) The maximum piece match error (maximum over all joint matches) called MAXERR.

A choice between two matches is based on these evaluations. The following tests are performed in the order presented here.

1. If the value of NEXCESS for two matches is different, then choose the match with lower value.
2. If the AVERR for one match is less than half that for the other match then choose the match with the lower value.
3. If the MAXERR for one match is less than half that for the other match then choose the match with the lower value.
4. Compute the ratio of AVERR for the match with the higher value to that with the lower value, call this AVRAT. Do the same for MAXERR, and call this ratio MAXRAT.

If AVRAT is larger than MAXRAT then choose the match with lower AVERR. Otherwise, choose the match with lower MAXERR.

In addition to the preferred match, this procedure also outputs a binary confidence rating for the choice. If the choice was made at either of the first three steps, then the choice is considered to be made with confidence, otherwise not. This confidence judgement was used in matching described previously to determine whether to keep the alternative matches.

Note that this selection procedure does not use any evaluation of the number of unmatched pieces in the model. It is assumed that occlusion can hide an arbitrary number of such pieces. Also the angles between different pieces were not used in making a choice. We have assumed arbitrary articulation of the limbs.

This selection procedure is designed to differentiate between two grossly different objects. Approaches for using model information to make finer choices are discussed in sec. 6.6.

APPENDIX 3 DATA STRUCTURES

We briefly describe the data structure of symbolic descriptions. Symbolic facilities of LEAP ([Feldman]) are used. Each Piece and joint of an object is an "Item", an atomic primary of LEAP. Each object is also an item. With each piece item we associate its summary descriptors as the datum of the item. The list of descriptors is the same as in sec. A2.1C of Appendix 2.

Joint items have a list of component pieces associated to them. Relations of joints and component pieces are in the form of the following LEAP "triples" (for the hammer of fig. 4.7):

$\text{RELATIONS} \circ \text{JOINT1} = [\text{COLLINEAR} \circ \text{PC1} = \text{PC2}]$

$\text{RELATIONS} \circ \text{JOINT1} = [\text{TYPE} \circ \text{JOINT1} = \text{T_JOINT}]$

Note in the above the relations in enclosed brackets are "Bracketed Triples". They are asserted as associations themselves. The relations used are: the different types of joints described in sec. 5.2, the collinearity, orthogonality and similarity of pieces.

Linking of two joints by a common piece is also stored as a triple, permitting easy traverse of the connection graph. E.g. for the doll in Fig. 5.2:

$\text{LINK} \circ [\text{LINKED} \circ \text{JOINT1} = \text{JOINT2}] = \text{PIECE1}$

This relation asserts that the JOINT1 and JOINT2 are linked and that the link is PIECE1.

Object descriptions are in terms of distinguished pieces as follows:

$\text{DIST_DESCRIPTIONS} \circ \text{DOLL} = \{ \text{DESCRIPTION1}, \text{DESCRIPTION2} \}$

$\text{DIST_PIECE} \circ \text{DESCRIPTION1} = \text{BODY}$

$\text{TYPE_DIST} \circ \text{DESCRIPTION1} = \text{WIDE_PIECE}$

$\text{REL_WIDS} \circ \text{DESCRIPTION1} = \{ \text{set of relative widths at two ends of main piece} \}$

In the above, a description item corresponds to each distinguished piece and indicated type of associations are made. One joint of the distinguished piece may have distinguishing features compared to the other, such as one joint has much longer pieces than the other, or has much wider pieces than the other. These are represented by the following type of assertions.

$\text{LONG_PIECE_JOINT} \circ \text{DESCRIPTION1} = \text{JOINTI}$

$\text{WIDE_PIECE_JOINT} \circ \text{DESCRIPTION1} = \text{JOINTJ}$

Isolated pieces have associated information about the closest joint and a possible connection, if more evidence for linking is available. E.g. for doll in Fig. 6.3:

CLOSEST_JOINT @ PIECE6 = JOINT1

CLOSEST_JOINT @ PIECE5 = JOINT2

POSSIBLE_CONNECTION @ PIECE6 = JOINT1

The last assertion is for a stronger hypothesis for connection, in our case generated by symmetry considerations (sec. 5.4).

REFERENCES

- [Alt], Franz L. Alt, "Digital pattern recognition by Moments" *Journal of the ACM*, Feb, 72, pp240-258.
- [Attneave] F. Attneave, "Some Informational Aspects of Visual Perception", *Psychol. Rev.*, 61, 183-193 (1954).
- [Agin 72] Gerald J. Agin, "Representation and Description of Curved Objects", Stanford Artificial Intelligence Project Memo AIM-173, October 72 (Ph.D. Thesis).
- [Agin 73] Gerald J. Agin and Thomas O. Binford, "Computer Description of Curved Objects", Third International Joint Conference on Artificial Intelligence, Stanford, Aug 73, pp. 629-640.
- [Ambler] A.P. Ambler, H.G. Barrow, C.M. Brown, R.M. Burstall, R.J. Popplestone, "A Versatile Computer-Controlled Assembly System", Third International Joint Conference on Artificial Intelligence, Aug 73, pp. 298-307.
- [Bajcsy] Ruzena Bajcsy, "Computer Description of Textured Surfaces", Third International Joint Conference on Artificial Intelligence, Aug 73, pp. 572-577.
- [Barrow] H.G. Barrow, A.P. Ambler, and R.M. Burstall, "Some Techniques for Recognizing Structures in Pictures", *Frontiers of Pattern Recognition* (ed. S. Watanabe), Academic Press, New York, 1972, pp. 1-29.
- [Baumgart] Bruce g. Baumgart, "Geometric Modeling for Computer Vision", Forthcoming Stanford Artificial Intelligence Laboratory Memo, 1974.
- [Binford] Thomas O. Binford, "Visual Perception by Computer", presented at the IEEE Conference on Systems and Control, Miami, December, 1971.
- [Blum] Harry Blum, "A Transformation for Extracting New Descriptors of Shape", Symposium on Models for Perception of Speech and Visual Form, Boston, November 11-14, 1964.
- [Bolles] Robert Bolles and Richard Paul, "The Use of Sensory Feedback in a Programmable Assembly System", Stanford Artificial Intelligence Laboratory Memo, AIM-220, October, 73.
- [Duda] Richard O. Duda and Peter E. Hart, "Pattern Recognition and Scene Analysis", John Wiley & Sons, 1973.
- [Earnest] Lester D. Earnest, "Choosing an Eye for a Computer", Stanford Artificial Intelligence Project Memo AIM-51, April, 1967.
- [Falk] Gilbert Falk, "Computer Interpretation of Imperfect Line Data as a Three-Dimensional Scene", Stanford Artificial Intelligence Laboratory Memo AIM-132, August, 1970.
- [Feldman] J.A. Feldman and P.D. Rovner, "An Algol Based Associative Language", *Communications of ACM*, August 1969, pp. 434-449.

- [Fischler] M.A. Fischler and R.A. Elschlager, "The Representation and matching of Pictorial Structures", IEEE Transactions on Computers, Vol. C-22, No. 1, Jan, 73, pp. 67-92.
- [Freuder 73a] Eugene C. Freuder, "Suggestion and Advice", MIT, AI Lab, Vision Flash 43, March 73.
- [Freuder 73b] Eugene C. Freuder, "Active Knowledge", MIT, AI Lab, Vision Flash 53, October 73.
- [Gill] Aharon Gill, "Visual Feedback and Related Problems in Computer Controlled Hand Eye Coordination", Stanford Artificial Intelligence Project Memo, AIM-178, October, 72.
- [Grape] Gunnar R. Grape, "Model Based (Intermediate-Level) Computer Vision", Stanford Artificial Intelligence Laboratory Memo AIM-201, May 73 (Ph.D. Thesis).
- [Gregory] R.L. Gregory, *The Intelligent Eye*, McGraw-Hill Book Company, New York 1971.
- [Guzman] A. Guzman, "Decomposition of Visual Scene into Three-Dimensional Bodies", Proceedings of AFIPS, 1968 Fall Joint Computer Conference, Vol. 33, Thompson Book Co., Washington D.C., pp. 291-304.
- [Hannah] Marsha Jo Hannah, "Computer Matching of Areas in Stereo Images", Stanford Artificial Intelligence Laboratory, AIM-239, July 74 (Ph.D. Thesis).
- [Horn] Berthold Klaus Paul Horn, "Shape from Shading: A Method for Finding the Shape of a Smooth Opaque Object from One View", Ph.D. Thesis, Massachusetts Institute of Technology, June, 1970.
- [Krakauer] L. J. Krakauer, "Computer Analysis of Visual Properties of Curved Objects", MAC TR-82, Ph.D. Thesis, MIT, May, 71.
- [Levine] M.D. Levine, D.A. O'Handley, and G.M. Yagi, "Computer Determination of depth maps", *Computer Graphics and Image Processing* (1973), 2, pp. 131-151.
- [Minsky] Marvin Minsky and Seymour Papert, "Perceptrons, An Introduction to Computational Geometry", The MIT Press, 1969.
- [Nevatia] R. K. Nevatia, "Depth measurement by Motion Stereo", Unpublished.
- [O'Niell] Barrett O'Niell, "Elementary Differential Geometry", Academic Press, 1966.
- [Posner] M.I. Posner, "Abstraction and the Process of Recognition", In G. Bower and J.T. Spence (Eds.), *Advances in Learning and Motivation*, Vol. III, Academic Press, New York, 1969.
- [Rivest] Ronald Rivest, "Analysis of Associative Retrieval Algorithms", Ph.D. Thesis, Stanford University, Computer Science Department, 1974.
- [Roberts 63] L. G. Roberts, "Machine Perception of Three-Dimensional Solids", Technical Report No. 315, Lincoln Laboratory, Massachusetts Institute of Technology, May, 1963.

- [Roberts 65] L. G. Roberts, "Homogeneous Matrix Representation and Manipulation of N-Dimensional Constructs", Document MS1045, Lincoln Laboratory, Massachusetts Institute of Technology, May, 1965.
- [Shirai] Yoshiaki Shirai and Motoi Suwa, "Recognition of Polyhedrons with a Range Finder", Second International Joint Conference on Artificial Intelligence, London, September 1-3, 1971.
- [Sobel] Irwin Sobel, "Camera Models and Machine Perception", Stanford Artificial Intelligence Project Memo AIM-121, May, 1970.
- [Tennenbaum] Jay M. Tennenbaum, On Locating Objects by Their Distinguishing Features in Multisensory Images, Technical Note 84, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, Calif.
- [Waltz] David A. Waltz, "Generating Semantic Descriptions from Drawings of Scenes with Shadows", Ph.D. thesis, MIT, AI TR-271, Nov, 72.
- [Widrow] Bernard Widrow, "The "Rubber-Mask" Technique-I. Pattern Measurement and Analysis" *Pattern Recognition*, 1973, Vol. 5, pp. 175-197. "The "Rubber-Mask" Technique-II. Pattern Storage and Recognition" *Pattern Recognition*, 1973, Vol. 5, pp. 199-211.
- [Will] P. M. Will and K. S. Pennington, "Grid Coding: A Novel Technique for Image Processing", *Proceedings of the IEEE* Vol. 60, No. 6, June 1972, pp. 669-680.
- [Winston 70] P.H. Winston, "Learning Structural Descriptions from Examples". Ph.D. Thesis, Massachusetts Institute of Technology, Jan, 70.
- [Winston 71] P.H. Winston, "Heterarchy in the MIT Robot", MIT Vision Flash 8.
- [Yakimovsky] Yoram Yakimovsky, "Scene Analysis Using a Semantic Base for Region Growing", Stanford Artificial Intelligence Laboratory Memo AIM-209, June 73 (Ph D. Thesis).